Un

(2)

26 August 1991

TPS TECHNOLOGY INNOVATIONS

Prepared for:
Naval Aviation Depot
Norfolk, VA 23511-5899

Prepared by:
Applied Research Laboratory of
Applied Research International
Virginia Beach, VA 23455

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	26 AUG 91	INTERIM

4. TITLE AND SUBTITLE
TPS TECHNOLOGY INNOVATIONS

5. FUNDING NUMBERS
C-N00189-90-C-0437
TA-A006

6. AUTHOR(S)
DAVID GORDON

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
APPLIED RESEARCH LABORATORY OF
APPLIED RESEARCH INTERNATIONAL
1300 DIAMOND SPRINGS ROAD
VIRGINIA BEACH, VA 23455

8. PERFORMING ORGANIZATION REPORT NUMBER
0437-A006

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
NAVAL AVIATION DEPOT
CODE 81300
NAS BLDG., LF-18
NORFOLK, VA 23511-5899

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
N00189-90-C-0437
CDRL A006

11. SUPPLEMENTARY NOTES
Second of Three Reports.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Publicly available.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report informs the reader about the important CAE software features required in the application of CAE software for TPS development. The report identifies an application methodology of CAE software to generate TPS information. Additionally, the report lays the ground work for identifying a CAE software application methodology for the DGAR's evaluation of the TPS design. The report discusses the application of CAE software tools for the Test Program Set (TPS) development process, CAE software selection criteria, and evaluation recommendations for CAE software tools. All TPS development tasks, from schematic entry to ATLAS code generation, are discussed and partially documented using a StaticInverter circuit for the study. Also, optimum CAE environment capabilities and features are identified for maximum TPS engineering productivity and efficiency.

The CAE study results reveal great potential in the application of CAE software to TPS development. The raw capability exists in several CAE environments which can be developed, with a few months effort into a CAE environment for TPS development.

(Cont'd.)

14. SUBJECT TERMS	15. NUMBER OF PAGES
Computer Aided Engineering (CAE) software, Test Program Set(TPS) Development, Analog simulation, Testability Analysis	61
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19 SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
6. AUTHOR(S)	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The study developed an analog fault simulation approach for the TPS development process. The study identifies a process to generate an analog fault dictionary which is essential for the testing analysis phase of TPS development.

14. SUBJECT TERMS	15. NUMBER OF PAGES
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

26 August 1991

TPS Technology Innovations

ARL Contract N00189-90-C-0437

Deliverable A006

TABLE OF CONTENTS

LIST OF FIGURES

EXECUTIVE SUMMARY

This report informs the reader about the important CAE software features required in the application of CAE software for TPS development. The report also identifies an application methodology of CAE software to generate TPS information. Additionally, the report lays the ground work for identifying a CAE software application methodology for the DGAR's evaluation of the TPS design. The report discusses the application of Computer Aided Engineering (CAE) tools for the TPS development process, CAE software selection criteria, and evaluation recommendations for CAE tools. All TPS development tasks, from schematic entry to ATLAS code generation, are discussed and partially documented using the SRA under study. Optimum CAE environment capabilities and features are identified for maximum TPS engineering productivity and efficiency.

Study results to date indicate that a total CAE approach for TPS development demonstrates great potential in TPS development productivity, efficiency, and quality. The analog simulator, Saber, is recommended as the analog simulation software because of the simulator's fault simulation and simulator modelling language capabilities. These capabilities allow the TPS engineer to generate an analog fault dictionary, which is an essential element to TPS development. Further recommendations are to select a CAE environment and the accompanying CAE software for TPS development.

The DGAR's needs(ie. to verify the quality of the developer's TPS design.) are different than that of the developer. Unfortunately, the TPS developer is not constrained by any particular TPS design tool or method and need only develop a TPS that conforms to all contractual specifications. Although the study covers CAE software that is applicable to TPS development, the study represents the first steps towards evaluating a methodology for use for the DGAR.

1.0 SCOPE

Applied Research Laboratory (ARL) is tasked with the evaluation of CAE tools and their application to the TPS development process. This report discusses primary evaluation results to date. Results discussed are the desired CAE environment capabilities, desired software tool's capabilities, and features and CAE software and hardware selection factors and issues. The CAE study covers a detailed application analysis of CAE software for the circuit simulation, fault simulation, and testing analysis tasks of TPS development. The results of the detailed application analysis are a demonstration of the proof of concept. The proof of concept is to propose the application of CAE software for TPS development.

2.0 INTRODUCTION

Technology innovations over the past decade have been a matter of evolution. Individual CAE and CAD software tools began as totally separate software entities. Individual IC (Integrated Circuit), system, circuit board, and TPS developers and manufacturers began using different combinations of CAE and CAD tools. Employing different combinations of CAE and CAD tools results in a requirement for the file format interface standards of EDIF (Electronic Data Interchange Format) and IGES (Inter Graphics Exchange Standard). Information was directly translated from one CAE tool into another CAE tool's information format. Today the individual CAE and CAD tools are merged into an environment which is a collection of CAE software tools. The normally separate design information from the CAE and CAD tools are part of a common database or unified databases. Individual CAE and CAD tools are assimilated or encapsulated into a specific vendor's definition of a "framework", "software backplane" or "environment". Selection of CAE and CAD software tools is an exercise in matching the designers needs with software tool capabilities, making capability or feature tradeoffs between available software tools, and selecting a CAE environment that contains the software tools.

This report discusses the critical capability and features required in the CAE software for TPS development. An empirical evaluation study process explores, applies, and investigates CAE software capabilities and features to the critical testability analysis and test strategy development phases of TPS development.

3.0 APPLICATION OF CAE SOFTWARE FOR TPS DEVELOPMENT

The purpose of CAE software application for TPS development is to generate the required information necessary to develop

the end TPS products, which are an ATLAS program and ID (Interface Device) hardware. Development of ID hardware(ie. electrical and mechanical design) is an obvious application for CAE tools, and as such is not discussed in this report.

The non-traditional application of CAE tools is to generate testing strategy information. A study of the informational requirements, informational flow, and the tasks to generate the testing strategy information for ATLAS program development is illustrated in Figure 1. This figure illustrates a total CAE software approach for TPS development. The figure relates the overall relationships between the separate testing strategy development tasks, the information generated and the end desired result, which is to generate the basic technical information (eg. testing sequences, ambiguity groups, testing voltages and testing tolerances) required to write an ATLAS program. An analysis of the information contained in the diagnostic ATLAS subsection reveals four essential pieces of information. The four information inputs are diagnostic tests and testing sequence, ambiguity groups, testing criteria at a test point, and testing stimulus. The following diagnostic tests, testing sequence, ambiguity groups, and testing criteria are derivative informational products of the fault dictionary that are discussed in the following sections of this report:

1. Monte Carlo analysis (Section 5.4).

2. Circuit simulation (Section 6.1).

3. Circuit modelling issues (Sections 6.2 through 6.6).

4. Fault simulation (Section 7.0).

5. Fault modelling (Section 7.3).

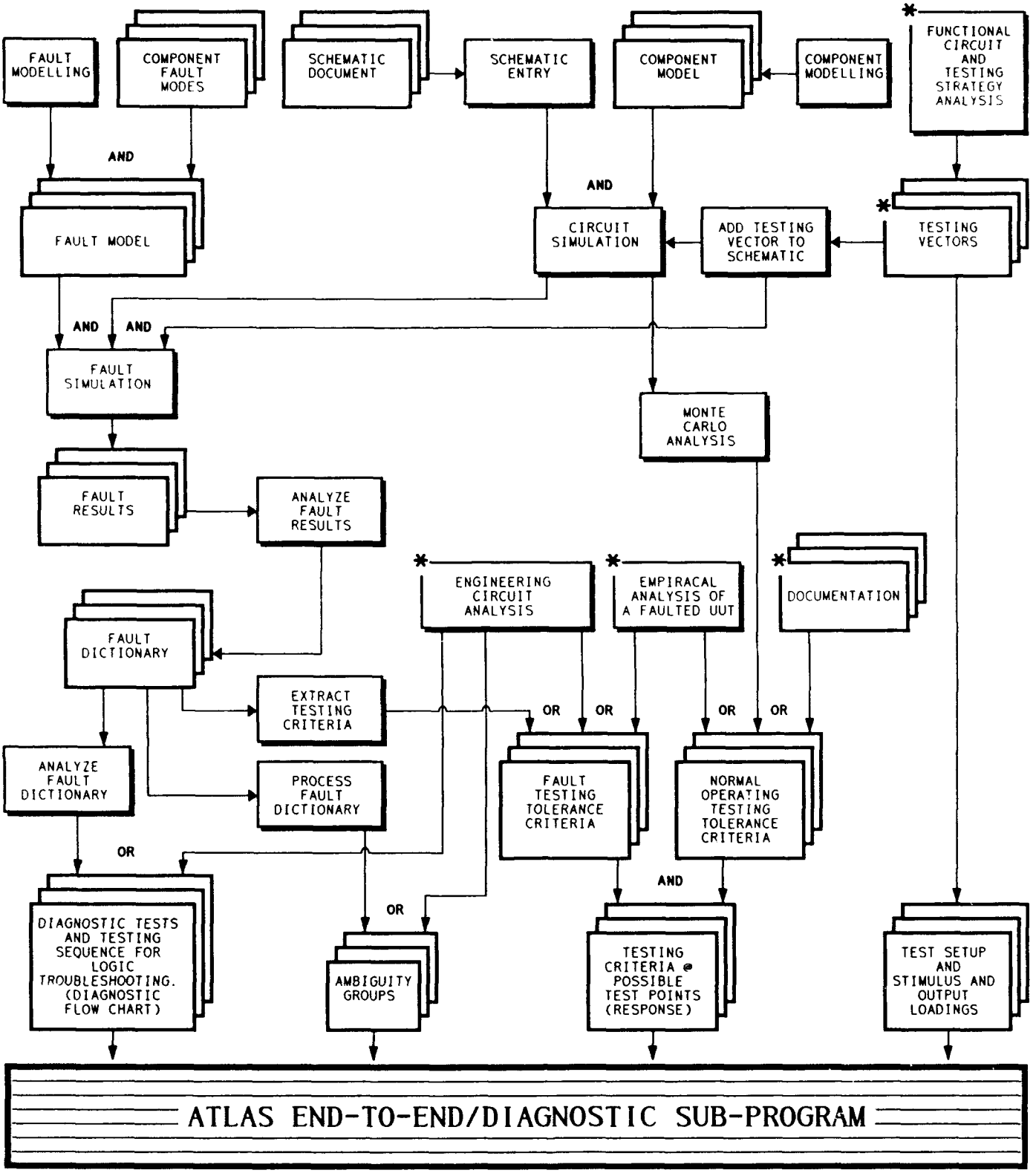
6. Fault dictionary and the analyze fault results task (Section 7.4).

7. Fault analysis, analyze fault dictionary, process fault dictionary, and extract criteria tasks (Sections 7.5 and 7.6).

The results of applying CAE software to the testing strategy, ID design, and documentation TPS tasks are the key focus points for the study of CAE utility and applicability for TPS development. CAE tools applied to ID design are not studied because the CAE tools studied are limited to electronic design. Application of CAE software tools to develop testing strategy is the major area for study, recommendation, and for development activities.

TASK AND INFORMATION INPUTS INTO THE ATLAS PROGRAM

FIGURE 1.



LEGEND

INFORMATION CONTAINED WITHIN THIS TYPE OF BLOCK IS CONSIDERED A TASK.

INFORMATION CONTAINED WITHIN THIS TYPE OF BLOCK IS CONSIDERED INFORMATION.

NOTE: INFORMATION FLOW IS FROM TOP TO BOTTOM.

* TRADITIONAL TPS TASKS

THIS DIAGRAM ILLUSTRATES TRADITIONAL AND FAULT SIMULATION APPROACHES FOR TPS DEVELOPMENT.

EACH TASK "CREATES" OR "TRANSMUTES" OR "ADDS TO" THE TASK INPUT INFORMATION INTO THE TASKS OUTPUT INFORMATION.

4.0 PRIMARY EVALUATION PROCESS

An empirical approach was employed to investigate the application of CAE tools for TPS development. After a preliminary evaluation, Cadence's Amadeus CAE software and Analogy's Saber software were selected for use. The Amadeus CAE environment provides the core set of CAE software tools for the basic schematic capture and digital simulation tasks. The Saber software tool provides the analog simulation capability. A task oriented circuit analysis and fault simulation process was conceived, implemented and evolved. Amadeus is the marketing product name for a standard set of software features, CAE programs and framework operating system environment produced by Cadence Design Systems. Amadeus includes a schematic drawing, design management, database library, and utility programs. Simulation of digital circuits were performed with another Cadence product named Verilog. Saber is an analog simulator produced by Analogy, Inc. Saber simulates analog and mixed mode circuits drawn in Amadeus.

Lessons learned during the application of the CAE software tools to the circuit and fault simulation tasks are discussed in Sections 6.0 and 7.0 of this report.

A secondary part of the primary evaluation process is the definition of the required CAE software tool capabilities and features specifically utilized for TPS development, identification of CAE software candidates, and selection for the implementation of the concept under study.

5.0 CAE TPS DEVELOPMENT ENVIRONMENT

5.1 OVERVIEW

CAE TPS development environment is a set of CAE software programs designed to work together for the purpose of TPS development. CAE TPS development environments do not exist. All available CAE environments are for the design of electrical circuits. The requirement is for a specific set of CAE software tools which work together (integrated) to provide TPS design specific information. The basic software set must include a schematic capture function, analog simulator with fault simulation capability, digital simulator with digital fault capability, and mixed mode simulator (a combination of the analog and digital simulators). The basic software set combination is necessary for the testing analysis and ID design phases. Additionally, the ID design task includes, a requirement for a CAD program (mechanical drawings), a possible requirement for a PCB router program, and a circuit simulator. For the documentation TPS task, all electrical and mechanical drawings produced by testing analysis and ID design must be transportable into a CALS compliant documentation program.

This report section discusses the CAE environment criteria, issues and capabilities necessary for TPS development. The study and "learn by doing" approach identifies the key features to minimize the users task of creating fault information required for testing analysis. Creating fault information required for testing analysis is the key goal for the application of CAE software for the TPS development task. Basic key features common to all CAE software were not specifically identified but are assumed because of the commonality between CAE software.

Independent stand alone testability analysis software does not exist, except for WSTA. Digital fault simulators provide few testability analysis parameter results. Integration of testability analysis software into the CAE environment is desirable, hence, the interest in exporting VHDL netlist representation of a circuit into WSTA. WSTA's strengths and weaknesses are discussed in Section 8.0 of this report.

The following subsections discuss CAE TPS development environment features, desired TPS CAE environment capabilities, analog simulator criteria, and CAE environment/vendor issues.

5.2 FEATURES

The purpose of the ideal TPS CAE environment and software tools are:

1. To generate the raw fault data and compute testability parameters (e.g. ambiguity parameters).

2. To process the fault data as much as is reasonably possible with a computer and provide the processed information to the TPS engineer.

3. To manage the TPS design information.

The currently available CAE environment products possesses the raw capability to accomplish all of the above requirements, except for the computation of testability parameters and the processing of fault data. CAE software does not exist to accomplish the above mentioned purposes in a ready to use (eg."out of the box") capability. Development of built in CAE features and auxiliary CAE data processing software is required to fully realize an ideal CAE based TPS development system. A workable CAE TPS development system can be created through development of the built in CAE features. Development of the built in CAE features enables the generation of raw fault data and management of TPS design information. The available CAE environments inherent deficiencies for the ideal CAE TPS development system are the computation of testability

parameters and raw fault data post processing. Development of testability parameter and fault data post processing capabilities are necessary but are not immediately realizable, because auxiliary CAE data processing software needs to be developed. A realizable CAE TPS environment would meet 40 percent of the ideal TPS CAE environment requirements. User development of the existing native capabilities is very practical and very cost effective when considering the degree of effort that is required to use CAE, if not developed.

5.3 IDEAL CAPABILITIES

A summary outline depicting the ideal TPS CAE environment capabilities is contained in this section. The outlined capabilities are the important capabilities distilled from the application and analysis of CAE software for TPS development. The intent of this outline is to identify criteria upon which to judge the completeness of a set of software tools contained in a TPS CAE environment. Each outline item is followed by a detailed justification or explanation. Items annotated with an asterisk (*) are not found in CAE software, but are desired to improve efficiency and productivity. Items annotated with a pound sign (#) are required for development of the inherent CAE capability. The rating or matching of these criteria to a specific CAE product is a future part of the study.

IDEAL TPS CAE ENVIRONMENT CAPABILITIES

1). Schematic diagram entry.

a.# Add user defined component parameters to the schematic database.

1) Modifiable databases allow the addition of component parameters unique for fault and test analysis tasks (un-detect field, MTBF field, fault set number, ambiguity group number, etc), user defined custom parts (hierarchial, functional, subassembly, SRA), simulation result parameters (pin or test point voltages or currents), or component parameters (resistances,etc).

b. Symbol creation.

1) Easily create symbols for functions, subassembly, SRA, standard hierarchical parts, connectors, and measurement functions.

c.# Simulator models in the schematic symbol library.

1) Need symbols and component parameters for all available components and behavioral models.

d.# Modify netlister instructions.

1) Creation of a hierarchial component or special component symbol requires a translation from the schematic component database field into the netlist syntax.

e.* Direct access to the behavioral analog and digital modelling language.

1) Need access to the component's simulator model from within the schematic environment without exiting or performing multiple actions to effect a model change. Need to constantly revise and work with the component model text during model development. Model development involves writing or borrowing existing models, then simulate, analyze results, then edit the model until the model is complete.

2). Simulate circuitry.

a. Digital simulation and digital fault simulation.

1) The digital simulator software must provide the essential fault results (fault dictionary) for TPS development.

b.# Analog simulation and analog fault simulation.

1) The analog simulator software must provide the essential fault results (fault waveforms) for TPS development.

c.# Mixed Signal and mixed fault simulation.

1) The mixed mode simulator must provide the essential fault results (both analog and digital fault information) for TPS development.

d. Multi-task the simulation.

1) For efficiency, schematic capture work and other computer tasks need to be performed, while a simulation is being run. Also, need

to execute multiple simultaneous simulation runs. These requirements help ensure maximum utilization of available computer resources.

e. Simulate on a remote network computer.

1) Need to run simulations on other workstation computers over the network. This ensures maximum utilization of available computer resources.

3). Integration between simulator and schematic capture.

a. Automatic display of user selected nodal waveforms.

1) This feature allows viewing of a group of waveforms during iterative schematic editing and simulation run tasks which saves time.

b. Simulator results annotated on the schematic.

1)* Waveform post processing results annotated on the schematic.

a) This feature assists the engineer in directly relating the essential simulator results to the schematic. Engineers think schematically.

b) Display the post processing waveform parameters such as RMS, average, peak, min, and frequency.

c. Post processing of simulation waveforms.

1) A majority of analysis time is spent extracting waveform parameters. Automatic extraction of waveform parameters would reduce the analysis task by a single order of magnitude.

a) Rise & Fall times for pulse shaped waveforms.

b) Marginal timing analysis for digital waveforms.

c) RMS, avg, peak, min for analog waveforms.

d) Would like to see automatic calculation of:

(1) Pulse parameters.

(2) Frequency parameters.

(3) General waveform parameters.

4). Test vector generation.

a.# Definable on schematic.

1) Create symbols and simulator models for standard testing vectors. Write a general simulator model which is modifiable for special test cases. Creating a testing vector symbol and writing a predefined testing vector as a component simplifies the fault simulation task.

b. Definable with test vector generation programming language.

1) Complex signal inputs, signals best described with a math function, and time repetitive signals are constructed with the native simulator modeling language. Writing a simulation model for the testing vector is necessary for a circuit simulation.

c.# Modelled test equipment components for simulation.

1) Models of frequency generators, arbitrary waveform generators, or power supplies are created. Test equipment characteristics and controlling parameters are modelled to create a soft instrument using the simulators modelling language.

d. Graphical or icon construction of the input vector.

1)* Implement on/off control of the testing resources. Power supplies are represented with a voltage or current source. Clocks with a clock component. Complicated test signals are created by connecting several sources together on the schematic.

5). Hierarchial modelling for analog and digital simulation.

1) Hierarchial modelling allows component, function, subassembly, or SRA substitution for different simulation comparison runs.

6). Documentation

a. Supports standard outputs of Postscript and HPGL.

1) Postscript and HPGL are universal formats for documentation programs.

b. CALS compliant

1) DXF - Mechanical Drawings.

1) CGM - Document Representation of Drawings.

2) EDIF - Electrical Drawings.

c. Generate the Engineering Support Data.

1) Simulation waveform capture for import or transfer into document program.

a) Allows documentation of fault results for TPS development and documentation. Need to create the graphical fault dictionary.

2)* Bill of materials.

a) Provides a report to document the components in a circuit.

3)* Customized report of user defined parameters.

a) Develop further undetectable/non-detectable component, failure mode and reliability reports. Customized report of user defined parameters provides a formatting tool to exchange component and TPS information contained in the schematic component database with other software programs. This subject is discussed in more detail in Section 9.3.

7).# Testability Analysis/Tools.

a.# Ambiguity group parameters.

1)# Automatic ambiguity parameter calculation. Testing strategy requires a testing sequence (a combination of tests defining ambiguity groups) that will meet the testability specification(ie. 80 percent of the components have to be in an ambiguity group of two or less). During testing strategy analysis, the TPS engineer adds and deletes diagnostic tests to change the ambiguity distribution to meet the specifications. Automatic ambiguity parameter calculation off-loads a tedious task from the TPS engineer.

b.# Testing Sequence.

1)# Provides the testing logic that is the structure for the diagnostic subtests of the test program. TPS quality is determined by testing sequence, testing tolerances, and fault dependency.

c.# Testing penalty data.

1)# Calculates the mean time, cost, and user value to isolate or repair.

5.4 ANALOG SIMULATOR CRITERIA

Described below are the predominate features required of an analog simulator for TPS development. These criteria and features are specific to the analog simulator software whereas the TPS CAE environment desired features are specific for the ideal CAE environment.

1). SIMULATION CONTROL

a. Analog simulator operation requires batch and interactive operational modes. Batch mode operation is needed to run unattended multiple simulations, thus providing for maximum utilization of machine and personnel. Lengthy fault simulation runs need to be ran overnight or over weekends. Interactive simulator operation is a necessity for an iterative investigation of circuit behavior and for operating the simulator.

2). MIXED MODE

a. Simulation of SRA digital and analog circuits can be accomplished with an analog simulator or a mixed mode simulator. Mixed mode simulations performed with an analog simulator is called a native mixed mode simulation. Native mixed mode simulation has simulation speed advantages over a mixed mode simulation whenever the number of digital gates is less than 300 gates. Also, simulation of digital and analog faults during the same simulation requires the faulted digital and analog components be simulated with the same simulator.

3). LIBRARIES

a. Simulator libraries are of single importance to lighten the modelling job. The more components characterized and modelled in the library, the less modelling is required, which eventually decreases the circuit simulation time. Need behavioral template for all component classes to provide a component substitute. Libraries should contain fault and behavioral components. Fault components are critically needed for fault simulation. Behavioral components or templates are required for modelling substitutions.

4). MODELLING

a. The importance of behavioral and hierarchical modelling cannot be overstressed. Modelling capability is essential to solve the problem of a non existent component model. Behavioral modelling provides a way to simplify and speed up simulation. Simplification of external driving signals and basic functional parts of the UUT not under immediate study is a great time saver because analog simulation takes a very long time. Analog behavioral languages provide a quick way to model components and develop a model substitute for the components that are not in the component library. A few hierarchial and behavioral advantages of hierarchical modelling are:

1) Different circuit configurations are easily generated by a substitution of a schematic symbol representing a substitute behavioral functional or assembly.

2) The behavioral substitution radically decreases simulation times (1X to 100X) over component level simulation.

3) Hierarchial circuit modelling provides for increased simulation efficiency which reduces simulation times of large circuit sizes.

Behavioral modelling language is required to correctly model failure effects during simulation. A fault simulation approach to develop circuit testing parameters requires behavioral modelling.

5). MONTE CARLO ANALYSIS

a. The Monte Carlo analysis task identifies the range of circuit electrical behavior due to the effects of varying component values. This technique determines the variance in circuit voltages and currents caused by variations in component values due to manufacturing or thermal differences. Manufacturing differences are due to all parts not being created equal or behaving exactly alike. Thermal differences between testing runs influence thermally sensitive components, thus causing different normal electrical outputs. Monte Carlo simulation is accomplished by the simulator statistically selecting different component values for each of the multiple simulator runs. The number of simulator runs has to be statistically significant. For example 100 simulator runs is statistically significant but 2 simulator runs is not. The comparison of the individual simulator results establishes the normal variance of the voltage or current of interest. The comparison of simulator results from the different Monte Carlo simulations by the engineer is the analysis part of the Monte Carlo analysis task. The Monte Carlo part of the Monte Carlo analysis task refers to the Monte Carlo simulation results.

5.5 CAE ENVIRONMENT/VENDOR ISSUES

1. FRAMEWORK

Framework is a marketing buzzword for CAE vendors to describe their CAE environment capabilities for their particular set of software tools. Framework refers to a software tool set providing a common user interface, and exchanging information between software tools in a CAE environment. There is one framework for each CAE vendor and one definition for each CAE vendor. The CAD Framework

Initiative (CFI) framework standard is several years away from possible market usefulness as a framework standard. This means a standard framework as a CAE selection factor is not is not possible. Additionally, some vendors add component database management, design management, and framework programming facilities. The software set of tools incorporated into a CAE framework and information exchanges between software tools (ie. integration) are the predominate criteria to judge CAE frameworks.

2. EDIF

The EDIF standard defines a file format to transfer design information between different CAE vendor programs. The information "stuffed" into the file is a netlist with schematic drawing information such as symbol shape and position included. The ability for an exchange of schematic drawings using EDIF involves the following criteria. The EDIF writer and EDIF reader parts of the two vendor programs must be compatible.

Theoretically, this should not matter with a standard file format, however implementations of the EDIF standard varies between vendors. Transfer of a schematic between any two specific CAE software programs is the only reliable method to test for schematic information transfer between vendors.

3. INTEGRATION

Integration of the software tool set is the most important factor in a CAE TPS environment, after an analog fault simulation capability. Simply stated, software integration is the degree to which information is exchanged between software programs.

Some key points about software integration are:

1. Software integration is a misunderstood CAE feature. CAE vendors exploit the obfuscation to define their own definition of integration.

2. Software integration possesses the most marketing hype of all CAE features because every vendor's definition is different.

3. Software integration has a significant impact on user productivity.

Integration between CAE software is the least understood and most discussed feature of CAE software. There are different degrees of integration between programs. The degree of integration is a factor of the completeness and mode of the information exchange. Information exchange completeness is how much of the information needed by the

receiver program is transferred by the supplier program. Information exchange mode is the transfer conditions under which the information is exchanged, i.e. automatically, conditionally or manually controlled by the user or a macro programming language.

Individual CAE software tools from different vendors are generally not well integrated with one another despite the marketing talk about standards. If a software tool is not part of the main set of software offerings, then the integration will be of the basic netlist transfer variety. Different levels of information exchanged between a schematic capture program and a simulator is defined as follows. Each increasing integration level possesses additional information exchange over the lower integration level. The more information exchanged and automatically transferred without operator intervention, the more efficient and productive the schematic and simulator program combination.

Level 1. Netlist transfer for the simulator. This is by far the majority of cases for SABER. Figure 2 shows this level of integration for the Amadeus to Saber interface. Amadeus is employed to draw the schematic and Saber performs the simulations.

Level 2. Netlist and simulator control commands are transferred into Saber automatically without operator manual intervention. (next version of the Amadeus to Saber interface).

Level 3. Netlist and simulator control commands are transferred into Saber. Simulation results are translated and displayed in the CAE environments waveform display software. Figure 3 shows this level of integration for the ComputerVision and Saber interface.

In addition to the above integration levels, direct editing of simulator models and the post processing of simulation results displayed on the schematic is a very valued integration feature.

Integration of Saber into a CAE environment entails providing for the information exchange between the CAE environment and Saber. The more information that is transferred between the schematic capture and the simulator, the more efficient and productive the TPS engineer can be during the schematic edit and simulation cycle. Figure 4 illustrates Saber's information inputs and outputs. Typically, only the Saber netlist is created and transferred into Saber. For complete integration, all inputs and outputs are exchanged between Saber and the CAE environment.

Amadeus and Saber
Integration Diagram

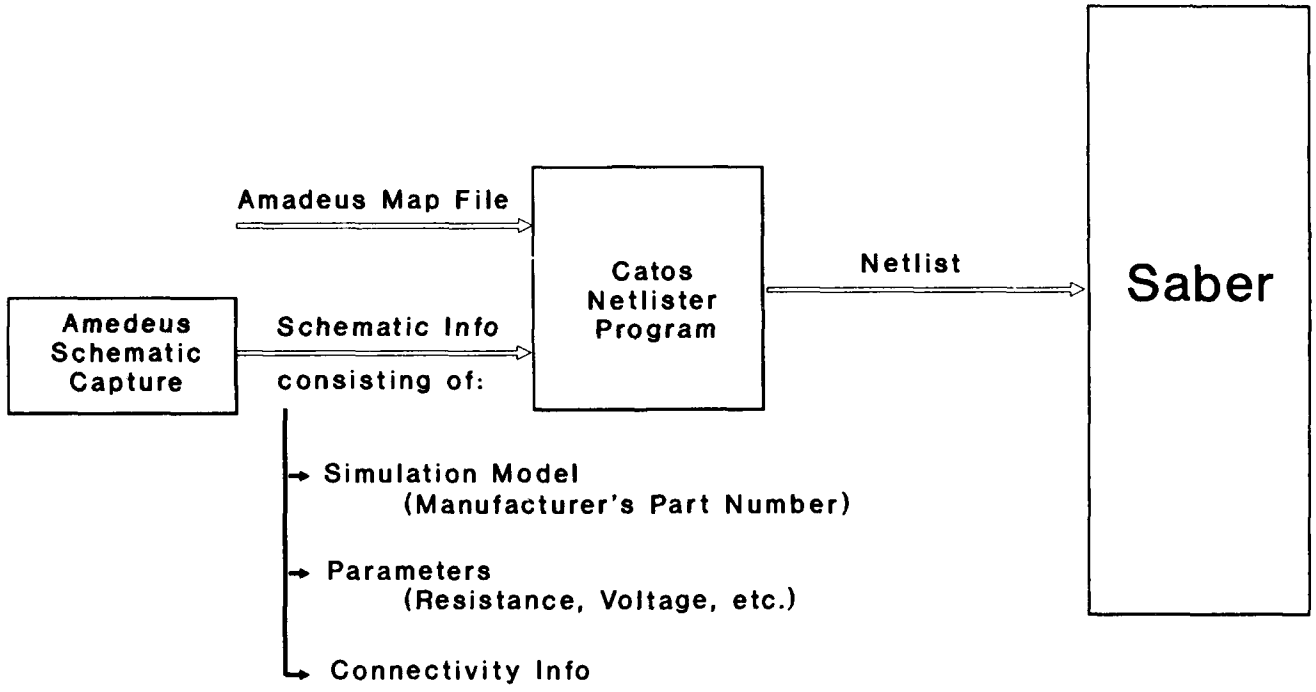


FIGURE 2

ComputerVision and Saber Integration Diagram

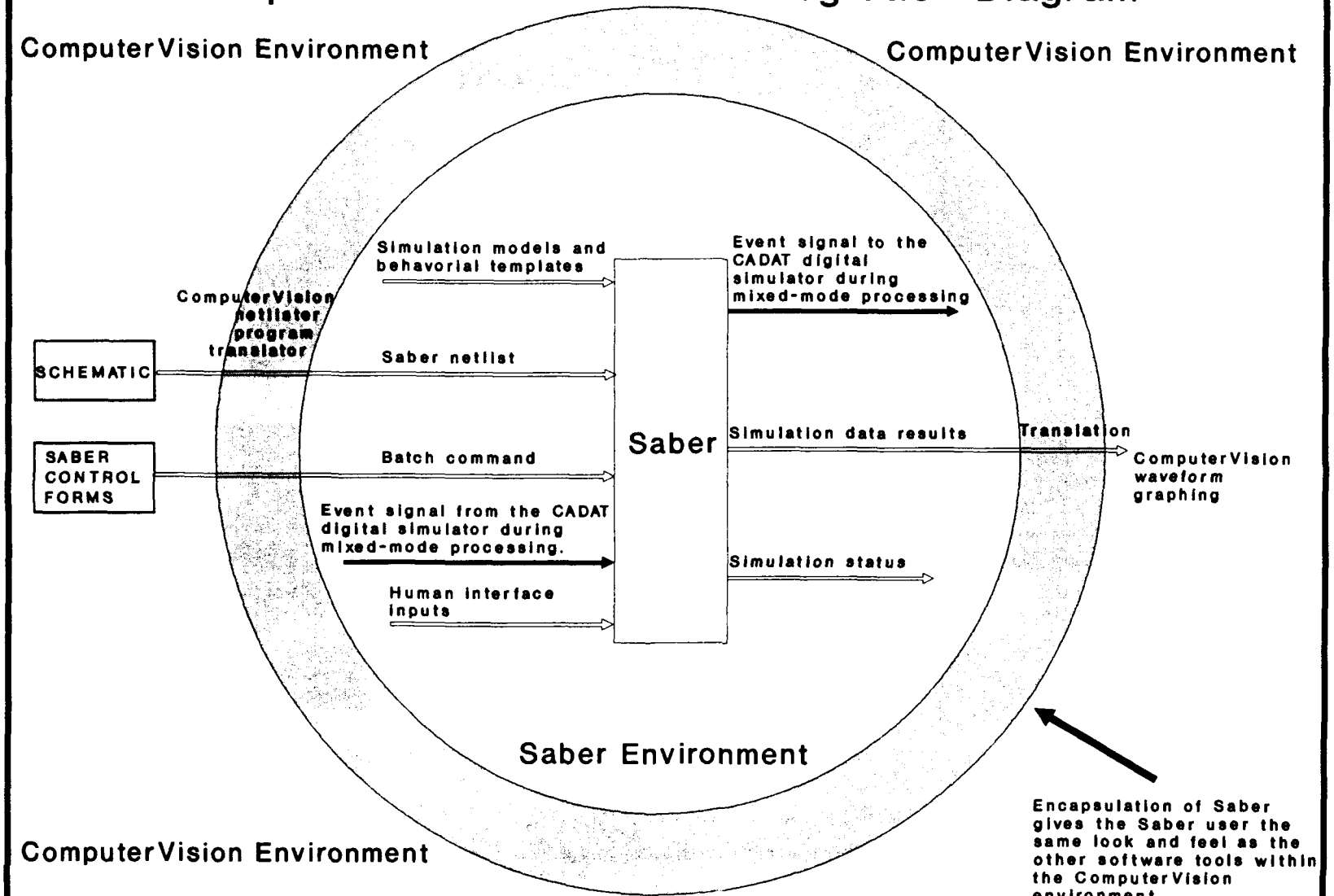


FIGURE 3

Saber Inputs and Outputs

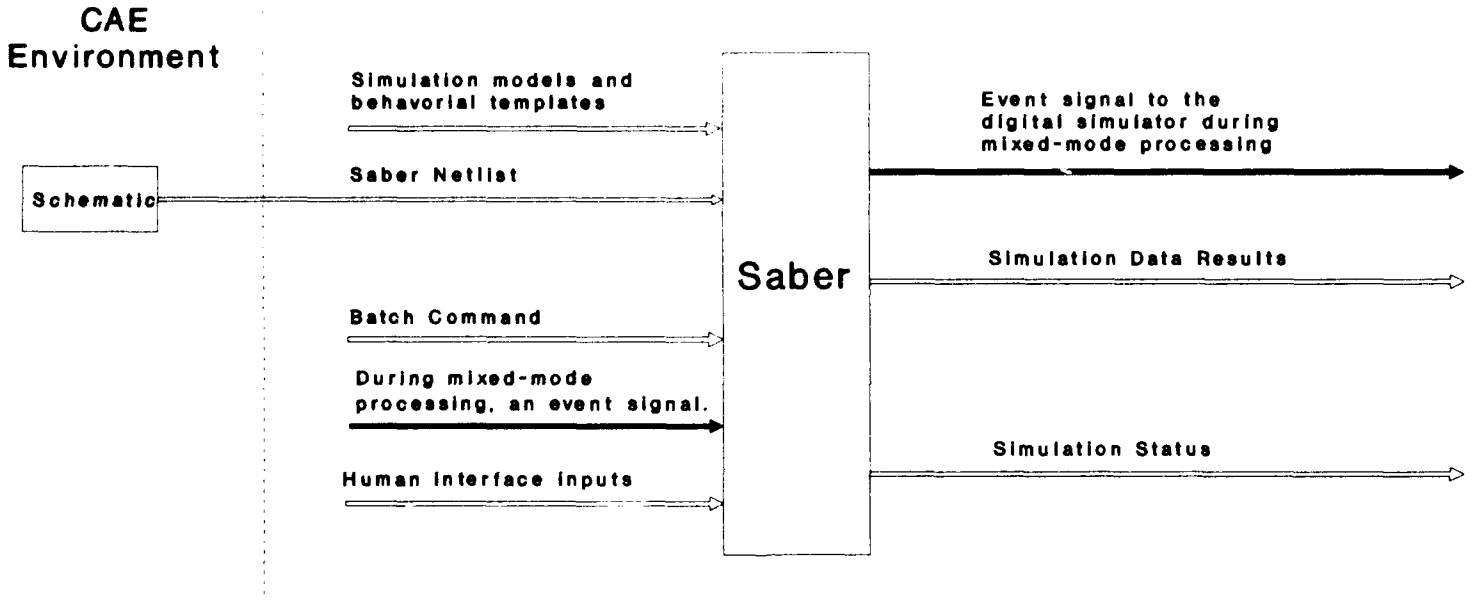


FIGURE 4

All CAE vendors except ComputerVision and Racal Redac have the basic netlist transfer level of integration. This level of integration presents a major time consuming obstacle for the circuit and fault simulation tasks. To date, the marketing survey reveals the ComputerVision has the best level of integration. ComputerVisions schematic drawing software interfaces with Sabers netlist, the batch command, and the simulation data results.

6.0 CIRCUIT SIMULATION

6.1 OVERVIEW

Circuit simulation task is a prerequisite for the fault simulation task. The circuit simulation task creates a circuit model that represents the circuit behavior during simulation. Fault simulation requires the circuit model developed during the circuit simulation task. Also, the circuit simulation task establishes a component organized information database and allows the TPS engineer to gain in-depth circuit knowledge, to develop end-to-end testing tolerances and diagnostic testing tolerances. Component information entry during the schematic drawing process organizes the component information to form a component database which is needed in documentation development, testability analysis, and TPS design management. The component database information is a valuable commodity because the same information is required for different TPS design activities which are accomplished months apart by different people. Thus, the component database serves as a master database for TPS design information. In-depth circuit functional knowledge and identification of critical design components are learned through the process of obtaining correct simulation results.

Variations in circuit electrical parameters under a range of normal operating conditions and in component manufacturing characteristics are not always specified in the TPS documentation. The normal or faulted electrical parameter variations require a Monte Carlo circuit simulation to vary the component electrical characteristics in a statistical manner. The range in test point voltages, currents, or resistances establishes the correct or normal testing tolerance for the test program.

A successful circuit simulation is an easily accomplished task if all the components are in the simulator's modelling library. For electronic designers, the components are in the simulation library, so circuit simulation proceeds smoothly. A TPS engineer does not possess the original design simulation library because of a non-simulator design approach (a breadboard prototype) or in some cases the older component technology (RTL digital technology), and the unique special component characteristics (pulse transformers

or hybrid circuits) required of military hardware. A point often overlooked is that a component in the schematic symbol library is useless to a TPS engineer if the component does not have a simulation model. Several real world facts that complicate circuit simulation are:

1. Well characterized(ie. accurate) component models do not exist for all active components.

2. Component substitution using idealized behavioral component models may cause a loss in simulation accuracy.

3. Analog simulation to the component level takes a very long time.

The following sections, Analog Modelling, Modelling Representation, and Simulation Design Cycle Time, correspond respectively with the above mentioned real world facts and address the real world complications for circuit simulation.

6.2 ANALOG COMPONENT MODELLING

Analog component modeling is the creation or selection of a model for a component, function, or system to achieve the desired simulation result. Simulation component models are chosen from a model library that best represents the specific aspect of the simulated component's behavior under study. A component simulator model existing in the component library obviously does not require component modelling for a standard simulation. If a component is not specifically characterized(ie not in the simulator model library), the next step is to identify a replacement that will give acceptable simulation results.

Acceptable simulation results depends on the engineer's intent which is to duplicate either the components functional behavior or exact electrical behavior under normal simulation conditions. Other than normal simulation conditions are Monte Carlo, age, temperature and fault effects. Engineering judgement is involved and is based on the application of the component (e.g. transistor for switching). The component characteristics are only good over specified or assumed ranges of operating conditions.

Similar component models may be modified with the manufacturer's specifications. Manufacturer's data alone is very often not sufficient to create a model. During ARL's evaluation, it was found that actual manufacturers data from IBM for 11 SRA components proved useful but the component characteristics were not complete enough for modelling purposes.

Alternatively, close behavioral modelling substitutes are often acceptable for component model substitution. Certain

component types, such as diodes are easily substituted with a behavioral model.

When the simulation results using component substitutes are unsatisfactory, then a more intensive job of measuring the essential component characteristics is needed to accomplish the modelling job. If the components characteristics are easily measured by test equipment (e.g. measure transformers inductance), then a simulation model is written using an analog modelling language (eg. MAST). Complex and exact electrical component characteristics require measurement on a specialized component tester. Component characterization to obtain the detailed data for modelling may be accomplished in-house or contracted out. Both options are costly, in either time or money. In-house characterization is the least costly but slows TPS development.

Figure 5 illustrates a flow chart of the modelling substitution decision points. The flow proceeds from a best to worst case model substitution solution. The best and worst case is judged by weighing the cost of component modelling in terms of manpower and money spent. The modelling diagram is a guideline to arrive at a modelling solution when a component has not been specifically characterized (i.e., not in the simulators component library). Each alternative modelling method in the diagram often involves an iterative simulation cycle to determine suitability of the model.

6.3 MODELLING REPRESENTATION

The purpose of simulation is to develop a working model of the circuit under development either during ID circuit design or during testing strategy development. A model is representative of a component, function or assembly behavior over a range of conditions. The simulation model is dependent upon the modelling effort required to ensure the accuracy of simulation results and for the derived information to have value. Selection of a model whose behavior is acceptable over a range of conditions is the key modelling task.

The basic law of modelling is that the greater the abstraction level of either a component's, a function's or a system's electrical behavior, the less information is available. In other words, specific detailed results cannot be distilled from general facts. Often, this modelling tenet is not well understood or appreciated because accurate voltages for testing tolerances are assumed to be derived from simple analog models.

Another basic modelling tenet is that an inverse relationship exists between model complexity and simulation speed.

ANALOG COMPONENT MODELLING

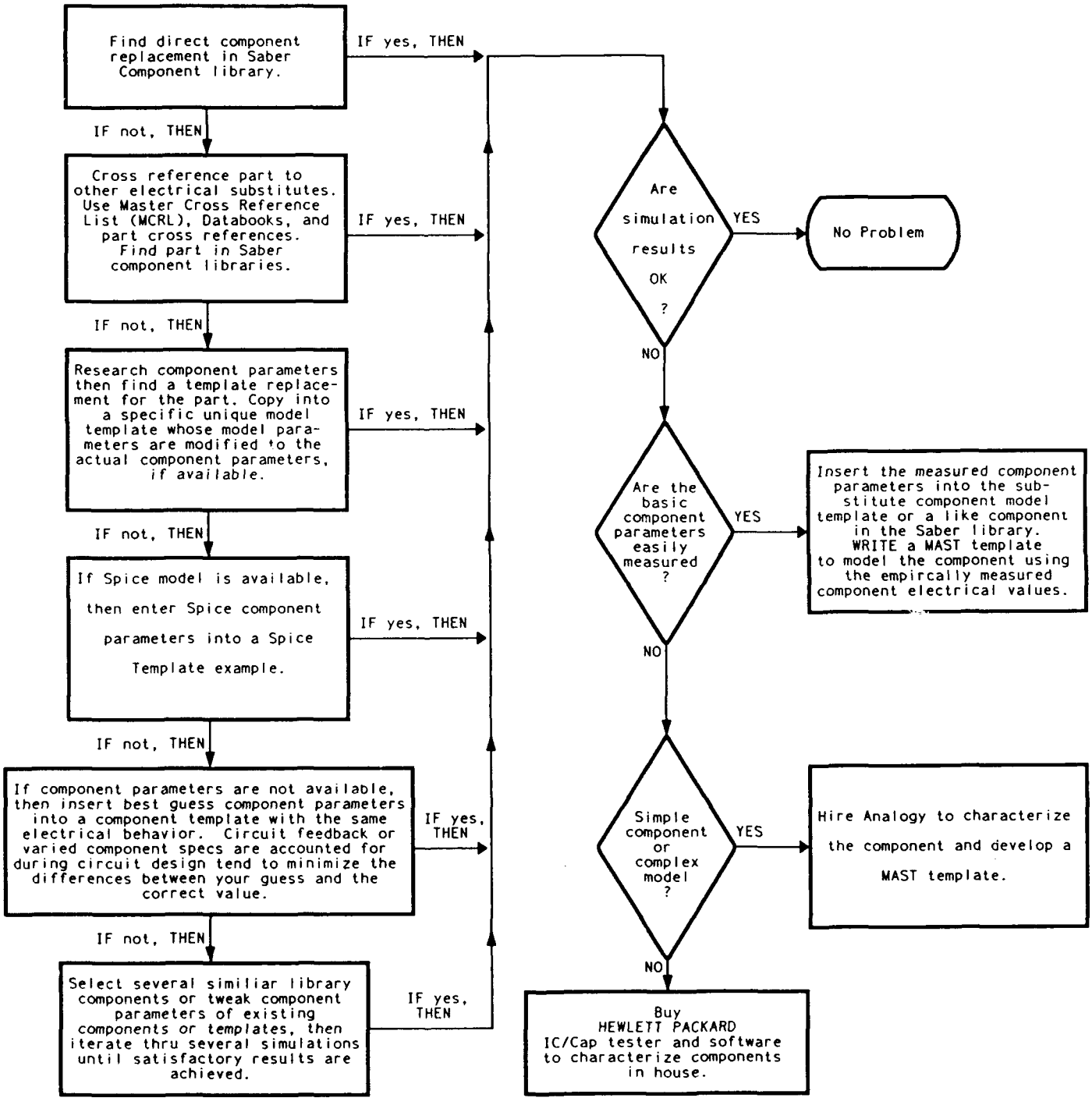


FIGURE 5.

Thus, more complex models result in decreased simulation speed and alternatively, less complex models result in a greater simulation speed. Substituting a model while maintaining good simulation is the significant endeavor in the modelling process.

The critical consideration about any circuit modelling (either component or behavioral modelling) is:

1. Are the simulation results (or information derived from the results) accurate enough or complete enough for TPS purposes?

TPS design requires establishing a dependency between a failure and the failure's effects on probable testing points, determining the failures effect on the circuit function, and determining a testing tolerance. The criteria for verification of simulation results are:

1. Are the fault results reasonable(ie. do the results correspond to the engineers functional and operational circuit knowledge)?

2. Are the fault results compatible with existing documentation?

3. Do the fault results correspond test cases where fault results are obtained from a real circuit?

Answers or explorative studies to these questions are required for full understanding of the modelling job and are addressed in the following paragraphs.

Modelling parameters considered in a model are: the mathematical relationship between input and output for both normal and failed conditions; manufacturing statistical variances in the parameters; and environmental influences such as temperature and age effects. Typically, only the relationship between input and output signals are modelled. Fully characterized commercial components have the mathematical input/output relationship and manufacturing statistical variances modelled. Environmental parameters are seldom modelled. Failure modes and age parameters are never built into the simulation model. The benefit of the information gained by modelling temperature and age do not justify the level of effort expended for TPS purposes. An analog modelling language simplifies the modelling of statistical and failure effects. If available, failure effects and statistical effects are required and desired modelling parameters for TPS development.

Behavioral modelling refers to modelling a component's basic behavior. All components in a component category have similar electrical characteristics. Behavioral models are a first order approximation of the standard component type

behavior. Differences from the average behavior are not modelled. The difference between a component model and a behavioral model is, the component model incorporates more modelling factors which more accurately reflects the component's actual performance.

A SPICE based definition of behavioral modelling is the ability to express an output quantity in the mathematical terms of the input. If a mathematical function accurately describes the components measured electrical behavior, then a behavioral model is perfectly substitutable for a component model. SPICE based simulators lack an analog modelling programming language.

An analog modelling programming language is essential to use for behavioral modelling in the TPS development effort. The Saber analog simulator features a programming language named MAST. All important component parameters for generic components are modelled in a template. Measured or manufacturing component parameters adjust the generic component templates to provide a specific component model. Saber is the only analog simulator to possess generic component templates for all component types, transistors, inductor, transformers, and gates.

Hierarchial simulation is a circuit simulation that employs models in functional or assembly sections. A hierarchical simulation approach allows the entire function, subassembly, or SRA to be represented as a singular block component in the schematic. Each functional or assembly section is a composite of the individual component models. Hierarchial simulation of the SRA or WRA functions and assemblies provides a way to substitute less ideal components(ie. behavioral model) for the functional circuit sections not under study while obtaining detailed simulation results. The behavioral modelling of substituted circuit functions and assemblies drastically decrease simulation times.

The hierarchial modelling approach allows for a mixed modelling of the circuit. A hierarchial mixed model approach is accomplished by substituting a detailed component model for the function or subassembly of interest, and then inserting behavioral models for all other circuit functions and subassemblies. The mixed modelling approach provides for detailed results while drastically reducing simulation time. Mixed modelling requires the behavioral modelling of the circuits functions, subassemblies, and SRAs which requires extra modelling time.

6.4 SIMULATION DESIGN CYCLE TIME

Building a simulation model of the UUT provides the basis for a fault simulation. Component, function, and subassembly models are constructed for the UUT. The time

required to build a UUT simulator model and perform a circuit simulation is dependent upon the following factors:

Direct factors affecting simulation times are:

1. Hardware.

    a) The main hardware specifications reflecting simulation speed is the computers rating in MIPS (Million Instructions Per Second) and MFLOPS (Million Floating Point Operations Per Second). The higher the number the faster the simulation.

2. The number of programs simultaneously running.

    a) Running any CPU intensive program at the same time as a simulation, steals CPU time from the simulation and prolongs the simulation time.

3. Modelling representation of the circuits components.

    a) The greater number of behavioral models used in the simulation, the less the simulation time.

4. Circuit Size.

    a) Simulation times for the Saber simulator is a linear function of circuit size. SPICE simulator times follow a quadratic function of the circuit size.

5. Component type.

    a) Non-linear components(ie. a non-linear transformer) require small time solution steps resulting in longer simulations.

6. Hierarchial circuit modelling.

    a) Hierarchial modelling allows for function or subassembly replacement in the simulation netlist. Substitution of behavioral models where appropriate, dramatically reduce simulation times. Hierarchial modelling reduces the simulation times (about 10-20 percent) for the Saber simulator.

7. Simulation control methods.

   a) An analog simulator solves large ordinary differential equation matrices incrementally in small time steps from the beginning to end of simulation time. The analog simulator employs an algorithm (or method of solution) to solve the circuit equations. With some circuits, the analog simulator is unable to solve the equations to the required accuracy using the default algorithm. This is called non-convergence. Adjustment of the simulators solution method and parameters allows the simulation algorithm to arrive at an acceptable answer (converge) which allows the simulation to proceed to completion. Adjustment of the simulators solution method reduces the accuracy and reduces simulation times. Another simulation control method is to restart a simulation at a stabilized or normal(eg. not during power up when transients are occurring) circuit operating state. Restarting a simulation at a normalized operating point eliminates start up transients which effects the time to do a simulation. This eliminates most of the simulation time for a complex power supply circuit.

8. Circuit type.

   a) If there are rapidly changing parameters (e.g. voltages), the simulators time step controller takes small time steps to achieve a solution. Taking small time steps increases the simulation time. A switching power supply simulation would take much longer than a full wave rectifier simulation.

Indirect factors affecting simulation time.

1. Integration of simulator and schematic editor.

   a. The more information exchanged between the schematic drawing and simulator, the less manual intervention by the user. If a schematic drawing program contains component symbols for all simulator models, then a hour and a half effort for symbol creation, component database parameter entry, and modification of netlister instructions is eliminated.

2. Behavioral modelling.

    a. Modelling provides the most significant improvement in simulation time over hardware factors. Improvement of simulation times, due to behavioral modelling can be up to a maximum of 100 times and an average of 10 times the normal component modelling simulation time. Detailed behavioral model application and substitution for a component model is desired to obtain the detailed information and simulation speed required.

3. Post processing of simulation results to extract the desired signal characteristics such as voltage, current, resistance.

    a. Post processing simulation results alleviates the engineer spending time interfacing with the waveform graphing program to extract the desired electrical parameters. Extraction of electrical parameters are the essential data required for test strategy development.

4. Machine/work schedule.

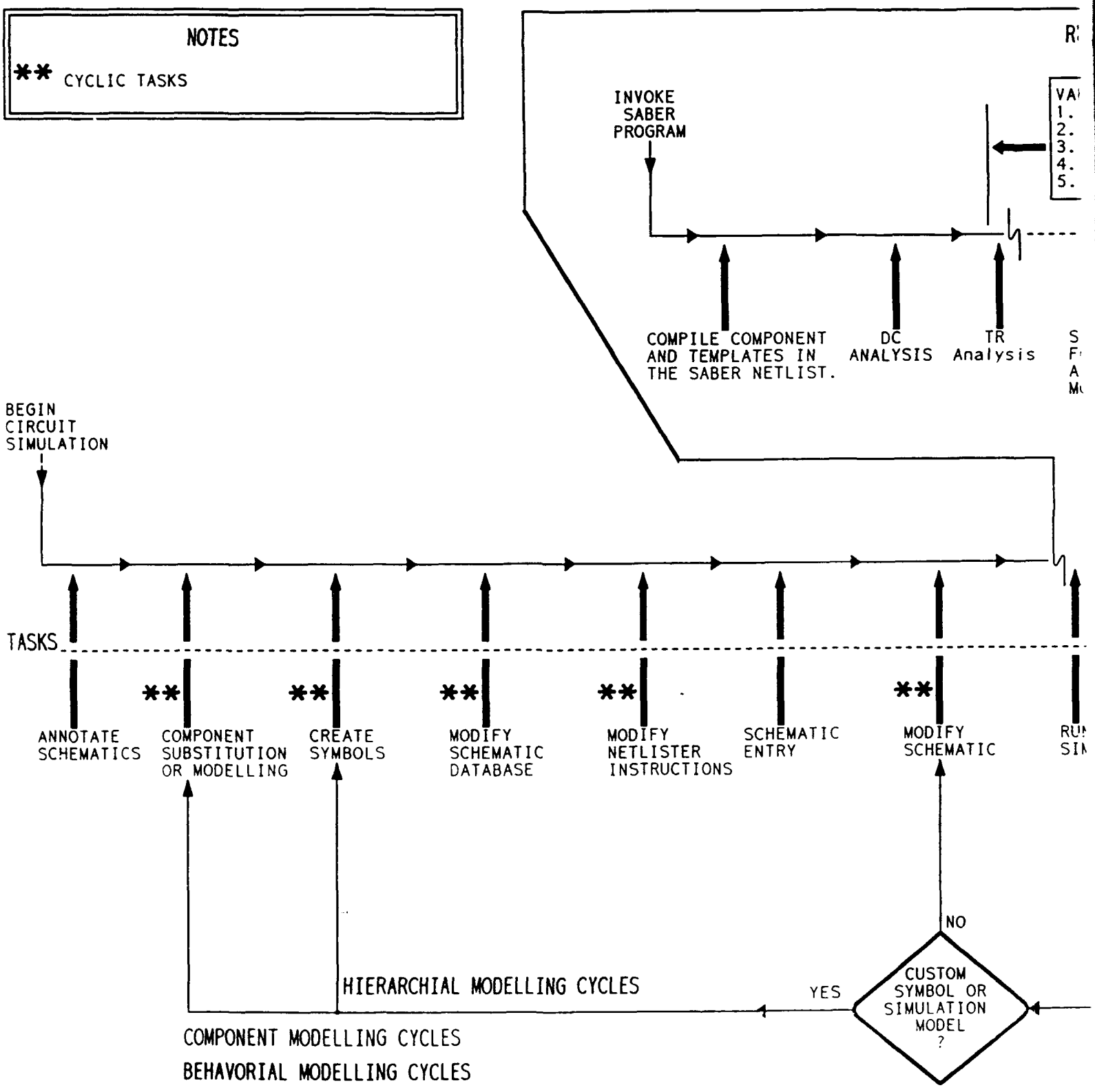
    a. Maximizing the utilization of resources can provide substantial improvements in productivity. For example: run unattended simulations overnight and on weekends, run simulations on unused machines over the network, or run a simulation while performing data entry functions such as schematic entry or word processing. The idea is to multi-task your work schedule and the computers work schedule.

5. Network resources.

    a. A fast workstation (e.g. the new HP 700 Series) on a network could be dedicated for simulation duty. Multiple simulation runs for Monte Carlo analysis could be ran on several workstations simultaneously, one simulation on each workstation.

Through a process of UUT modelling and simulation, basic tasks were identified. Figure 6 illustrates the basic simulation tasks and events.

CIRCUIT SIMULATION TA



.ATION TASK PROCESSES

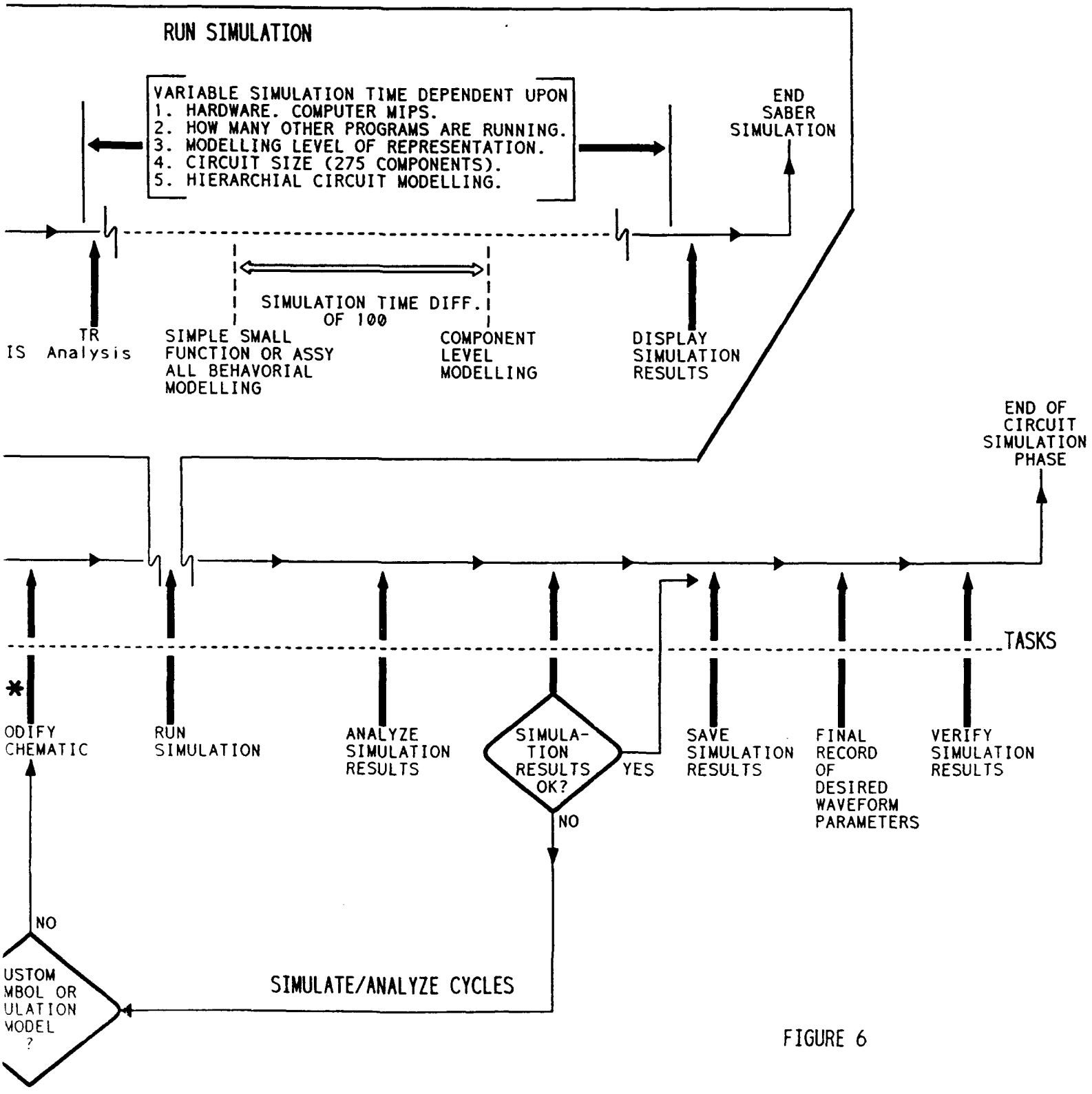


FIGURE 6

Four cyclical major tasks are:

1. Basic simulate/analyze cycle.

2. Hierarchial modelling cycle.

3. Component modelling cycle.

4. Behavioral Modelling cycle.

The circuit simulation task terminates when the circuit simulation results are compared with the waveforms of the actual circuit. A less optimum verification is to compare simulation results with the electrical waveform parameters recorded in the TPS documentation such as input and output specifications or VITAL listings. An alternative verification is to use the engineers functional, operational, and electrical knowledge gained from studying the schematic and from the simulation task to verify simulation results.

6.5 WRA MODELLING

WRA modelling may be done at the component level or behavior level depending on circuit size and the time versus accuracy tradeoffs associated with the component or behavioral modelling. Whether the simulation results or information derived from the results are accurate or complete enough for TPS purposes is the critical consideration about any circuit modelling (either component or behavioral modelling). In the real world, long analog simulation times for "large" circuit sizes dictates a behavioral model approach for WRA modelling. The definition of a "too long" simulation time is dependent upon the comparison of simulation time to the available time and manpower. Accepting 8 hours as the time limit for one simulation run, an analog and digital component count of greater than 300 to 500 analog components is the limit before any behavioral modelling is mandated or considered. The rough correlation, 500 analog components = 8 hours simulation time, is based upon an actual simulation time and is dependent upon the factors cited in the previous discussion of Simulation Design Cycle Time. Maximum digital simulation times have not been studied.

6.6 CIRCUIT SIMULATION TASK

Circuit simulation tasks and times to accomplish those tasks were recorded during the application study. Task times are annotated on Figure 7, Circuit Simulation Task Times.

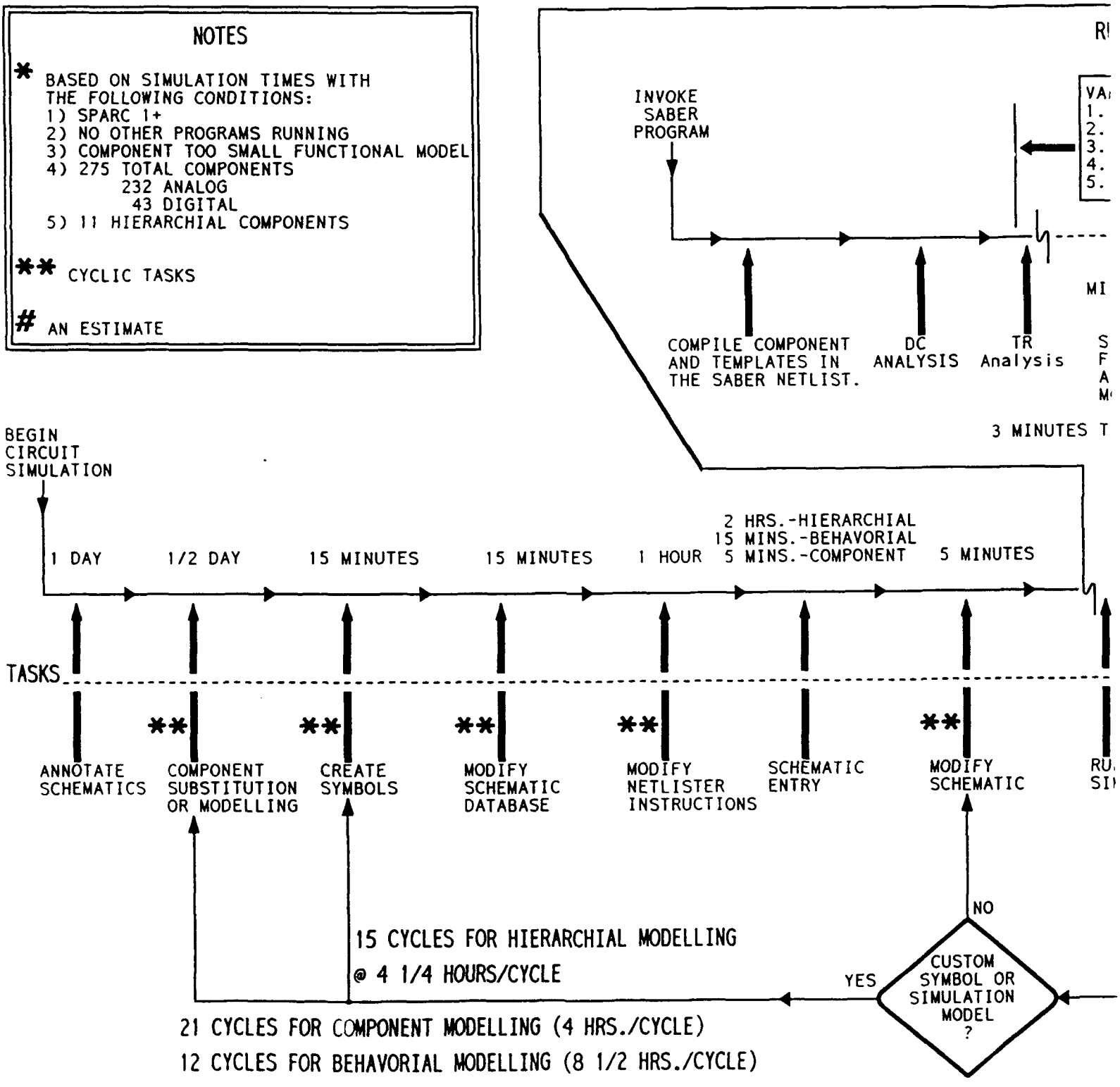
CIRCUIT SIMULATION ESTI

NOTES

* BASED ON SIMULATION TIMES WITH THE FOLLOWING CONDITIONS:
1) SPARC 1+
2) NO OTHER PROGRAMS RUNNING
3) COMPONENT TOO SMALL FUNCTIONAL MODEL
4) 275 TOTAL COMPONENTS
232 ANALOG
43 DIGITAL
5) 11 HIERARCHIAL COMPONENTS

** CYCLIC TASKS

\# AN ESTIMATE



ON ESTIMATED TASK TIME

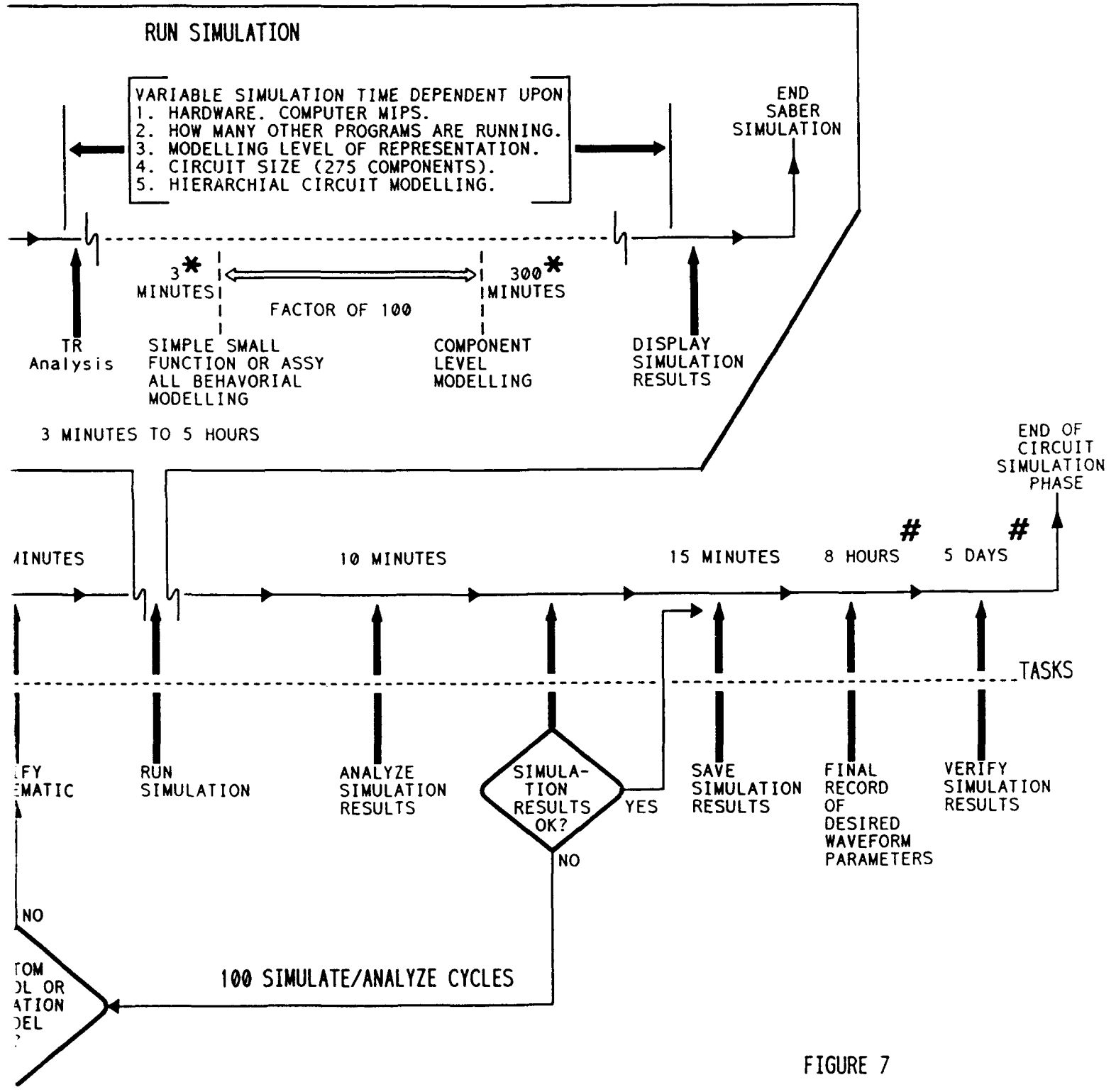


FIGURE 7

Conclusions derived from the circuit simulation task analysis are:

1. An estimated total circuit simulation time is 42 days.

2. Entry of circuit design information takes a third of the total time.

3. Modelling and model verification takes a third of the total time.

4. Analog simulation takes a third of the total time.

5. Circuit simulation is highly iterative.

7.0 FAULT SIMULATION

7.1 OVERVIEW

Fault simulation generates the raw data for development of a fault dictionary. The raw data establishes the precise dependency between a fault and the fault symptom, a key task which previously was highly dependent upon the TPS engineer's experience, a variable quantity. Fault simulation leads to the development of the fault dictionary. The fault dictionary is the essential TPS design database for TPS development.

The testing strategy process creates a fault dictionary, formulates a testing strategy, organizes a diagnostic testing sequence, calculates testability parameters, and ensures conformance of the testability parameters with the testability specifications. The fault dictionary provides the raw material for the establishment of a voltage, current, or resistance testing criteria for a diagnostic decision point.

Analysis of the fault dictionary establishes the dependency and relationship of a faulted component to a corresponding measurable fault symptom. Establishing the dependency and relationship of a fault with an observable fault symptom is a key task of the TPS engineer. In the past, the TPS engineer's knowledge of circuit behavior through physical fault insertion, established the relationship between a faulted component and fault symptom. Establishment of rudimentary circuit analysis of cause and effect relationships and an approximation of fault voltages, current, and resistance values were the best the TPS engineer could do. Voltage, current, and resistance measurements of an inserted fault on the UUT provides precise testing criteria for that particular SRA but not for other UUTs due to slight variations in component parameters. The precise testing criteria ensures the test program will correctly test different UUTs of the same type. However,a

traditional TPS development approach is limited to non-destructive singular faults that will not overstress other components, physically damage a board, and time limitations due to fault insertion quantity.

Fault simulation provides many advantages over traditional circuit fault analysis methods. Typical fault simulation advantages are:

1. Precise cause and effect dependency and relationships between the fault and the fault symptom.

2. Provides data to establish precise fault testing criteria.

3. Capable of establishing fault data for all components.

4. Capable of simulating the insertion of destructive faults (without damaging the actual equipment) to obtain fault data.

5. Simulate "real world" faults versus the idealized open/short faults.

Simulating faults that are commonly experienced ensures testing program quality through the ultimate benchmark - "Can the test program find 'real world" faults?" For the most part, the idealized faults are representative of real world faults. Often, MOSFET and CMOS component failures do not exhibit the idealized short or open characterization. For example, metal oxide semiconductors exhibit kilo-ohm resistances as a failure mode instead of an open. Modelling a component failure during overvoltage or overcurrent conditions further provides a mechanism to predict secondary fault effects which are typically overlooked. The overstressing of an associated component leads to failure of the second component during actual fleet use and often is the recurring cause of the same component failing repeatedly. Discrete diode replacement procedures in a full wave rectifier configuration is an example of the phenomena just described. Another example is the case where a feedback resistor is defective and causes an OPAMP to fail. The failed resistor is not discovered and the OPAMP is replaced. The resistor is the primary cause of the OPAMP failure, not the OPAMP itself. The secondary effect of the Out of Tolerance (OOT) resistor fault is to cause the failure of the OPAMP yet he OPAMP appears to be the single point of failure.

7.2 FAULT SIMULATION TASK

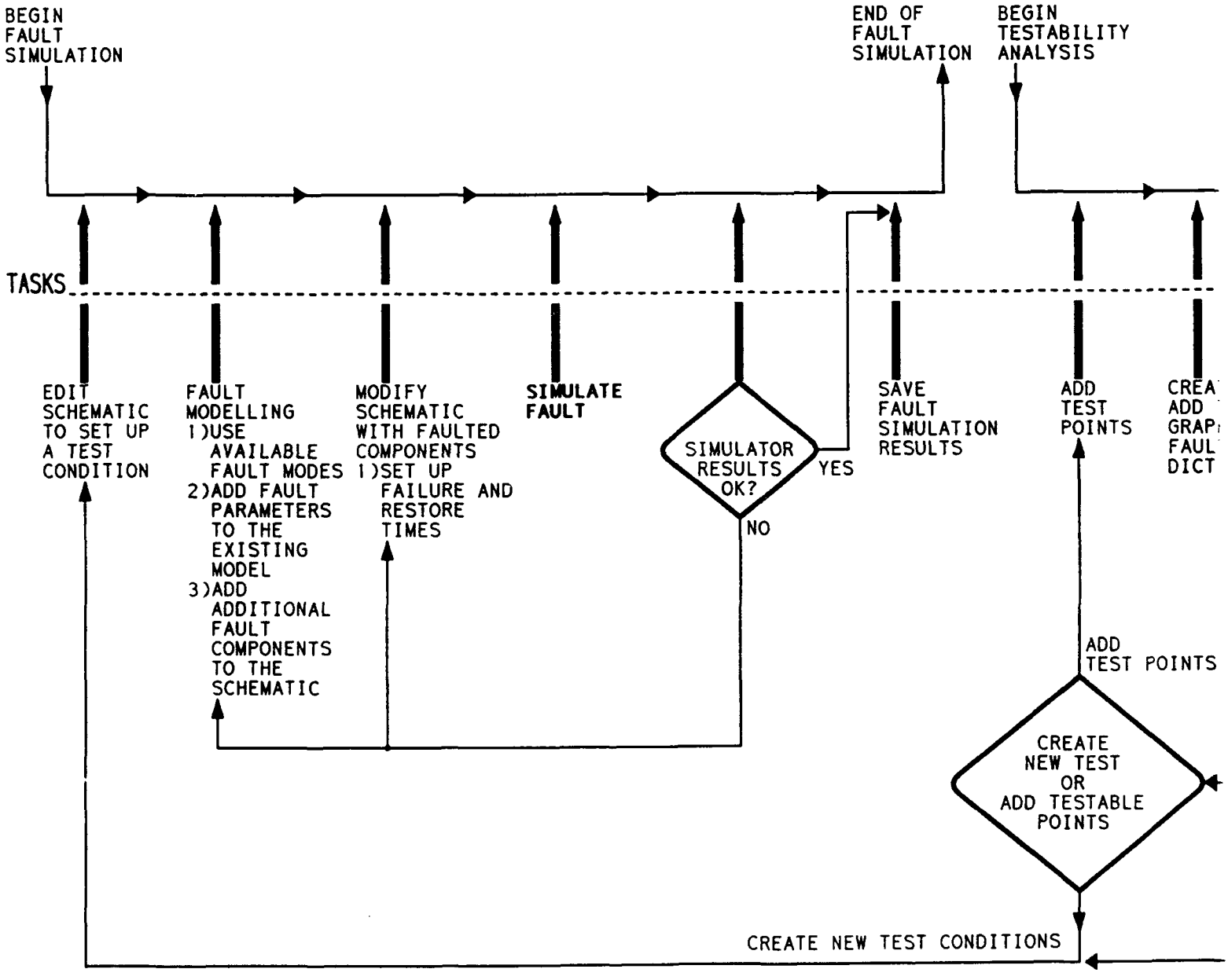
The fault simulation task involves configuring a circuit's test condition, fault modelling, and simulating faults. Fault simulation results are analyzed and converted into a fault dictionary. Test condition configuration is determined by the engineer from documented testing requirements or operational knowledge about the circuit. Fault modelling is accomplished by using available fault models (e.g. resistors, capacitors, diodes), adding fault parameters to the existing model (e.g. transistors), or adding extra components (e.g. a short or open component) to the schematic. A group of failures are simulated at different times in the simulation but never at the same time during the simulation. This procedure saves simulation time and simulates fault component groups in a function or subassembly for fault analysis purposes. If the fault simulation is satisfactory, then all waveforms are saved into a different directory. Figure 8 illustrates the fault simulation task sequence. The fault simulation task is part of an iterative task to simulate the fault behavior of all components. Low MTBF components are faulted and simulated individually or in a group if possible. Then all active components are simulated individually or in a group, if possible. Finally, passive components in each functional group are fault simulated.

7.3 FAULT MODELLING

Fault modelling entails changing the normal components electrical behavior to a faulted component electrical behavior. The behavior change is made internally to the components simulator model or externally by placing an external component (a short or open component) in the circuit adjacent to the faulted component. Generic standard components, such as resistors, are fault modeled by simple model substitution, either by editing the schematic or directly editing the netlist. Complex components, functions, or subassemblies require a dedicated modelling effort.

A modelling approach is to characterize a batch of failed components through measurement of their electrical behavior. Next, the faulted electrical data is either fitted to a function or directly modelled by Saber's modelling language. Partially failed components could likewise be modelled. Most failures are approximations to the idealized fault modes delineated in MIL-STD-217 or in the CASS contract. Logistical data identifies commonly failed components but lack the specific details on the component fault behavior. Manufacturing quality assurance data is a possible source of fault characterization, however the data needs to be from the actual operational environment.

FAULT SIMULATION AND TESTABI



'ESTABILITY ANALYSIS TASK PROCESSES

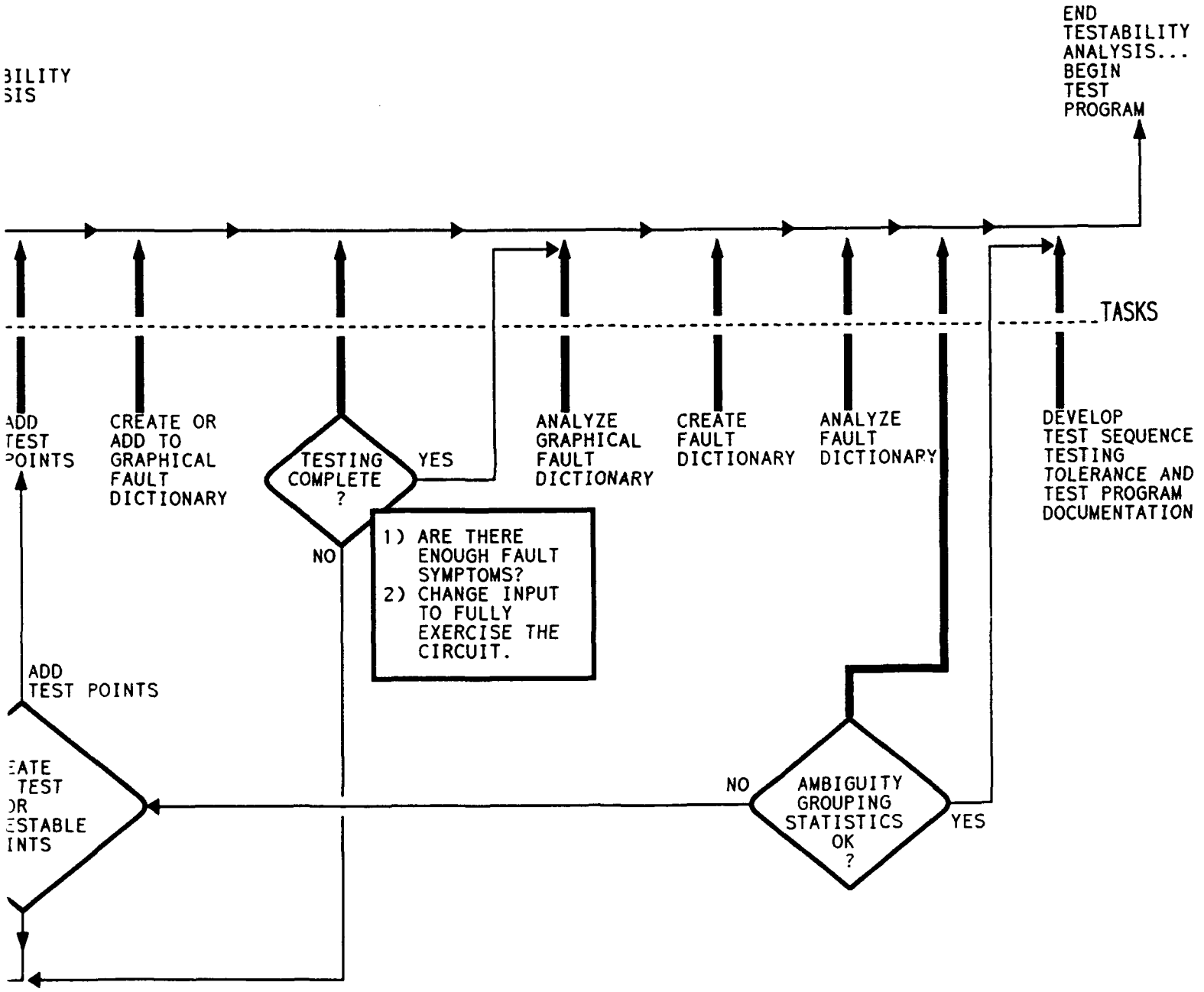


FIGURE 8

A collective survey and consensus from experienced field engineers is a valuable source of fault information.

Devices possessing two terminals (ie. two port devices) are easily modelled by shorting the input to output or by opening the input to output circuit path. An alternative method is to change the components main characteristic parameter which has simulation speed advantages. For example a resistor from 0 ohms to 1 teraohm or a capacitor from 1 megafarad (short) to 1 femtofarad (open).

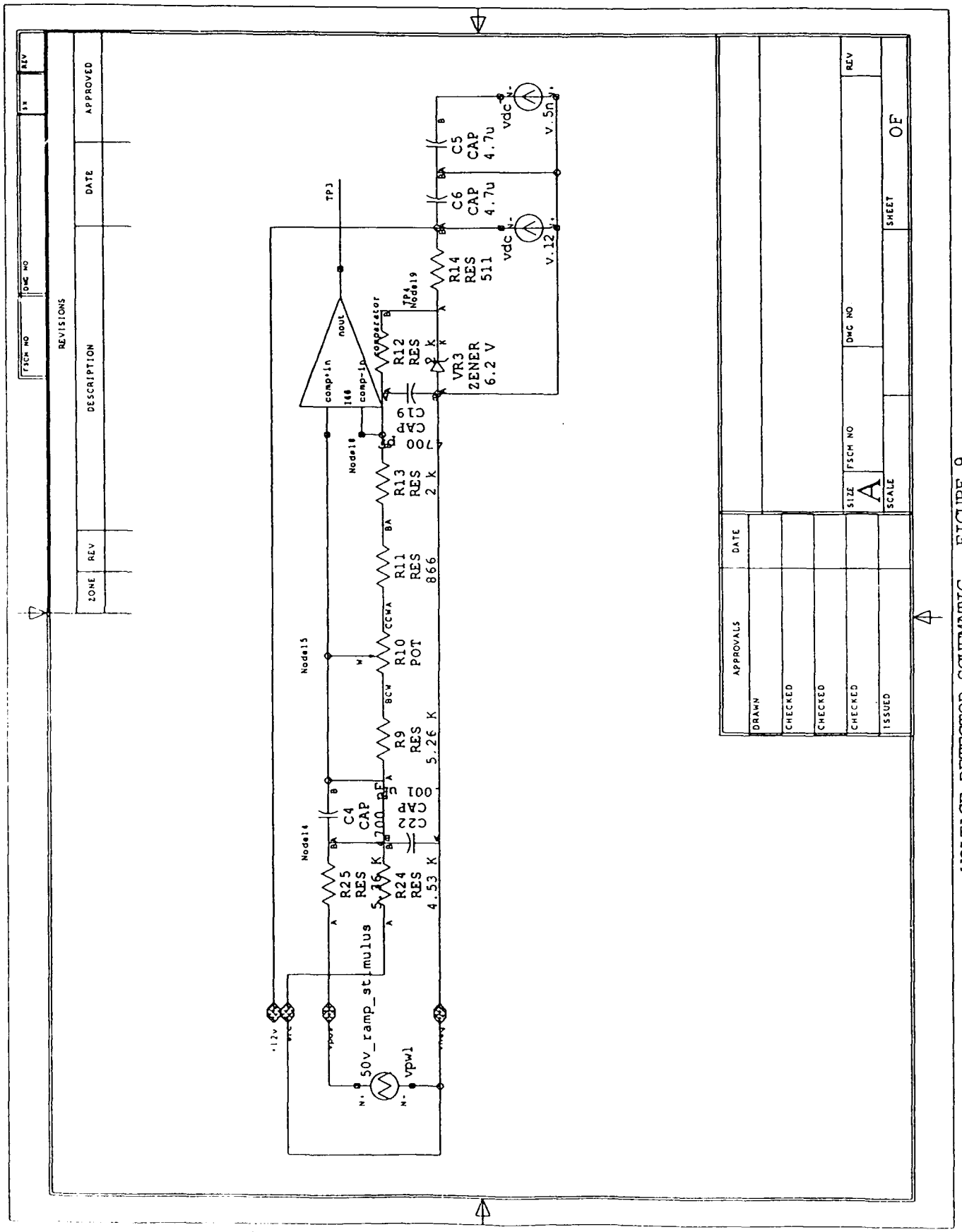
An area for study is the effect of a fault on other components. Component models are often modelled specifically to operate under normal conditions. A voltage or current from another faulted component may place the component simulation in a poorly characterized region. Simulation in a poorly characterized region results in poor component behavior and presents a simulation accuracy problem.

Saber fault simulation is accomplished by changing the component electrical characteristics at a specific simulation time or when a specified voltage, current, or power dissipation event has occurred during the simulation. Typically, many components are set up to fail then recover from failure during a simulation run. The "time fail and heal" method generates the largest amount of fault symptoms corresponding to the faults over single failure/simulate methods.

7.4 FAULT DICTIONARY

The fault dictionary is a compendium of fault analyses that relate fault conditions to corresponding fault symptoms. Relating a fault to a precise fault symptom and establishing the precise dependency is the "golden egg" required for diagnostic accuracy of the TPS.

Derivation of the information in the fault dictionary is from the fault simulation results. The simulation results are organized in a graphical format. The graphical format is called a graphical fault dictionary. Components of the Voltage Detector circuit in Figure 9 was faulted and the resulting graphical fault dictionary is portrayed in Figure 10. The Voltage Detector circuit resistors and capacitors were failed as shorts, one at a time sequentially during a Voltage Detector simulation. The electrical parameter of interest (eg. current or voltage) was recorded for each testable or probable circuit node. The recorded results are illustrated in horizontal Sections 2 through 9 of the Graphical Fault Dictionary. Each 10 milliseconds is a fault time bin representing a different fault condition of the Voltage Detector at all testable points. For every 10 milliseconds, from 0 to 130 milliseconds, a different component was failed.



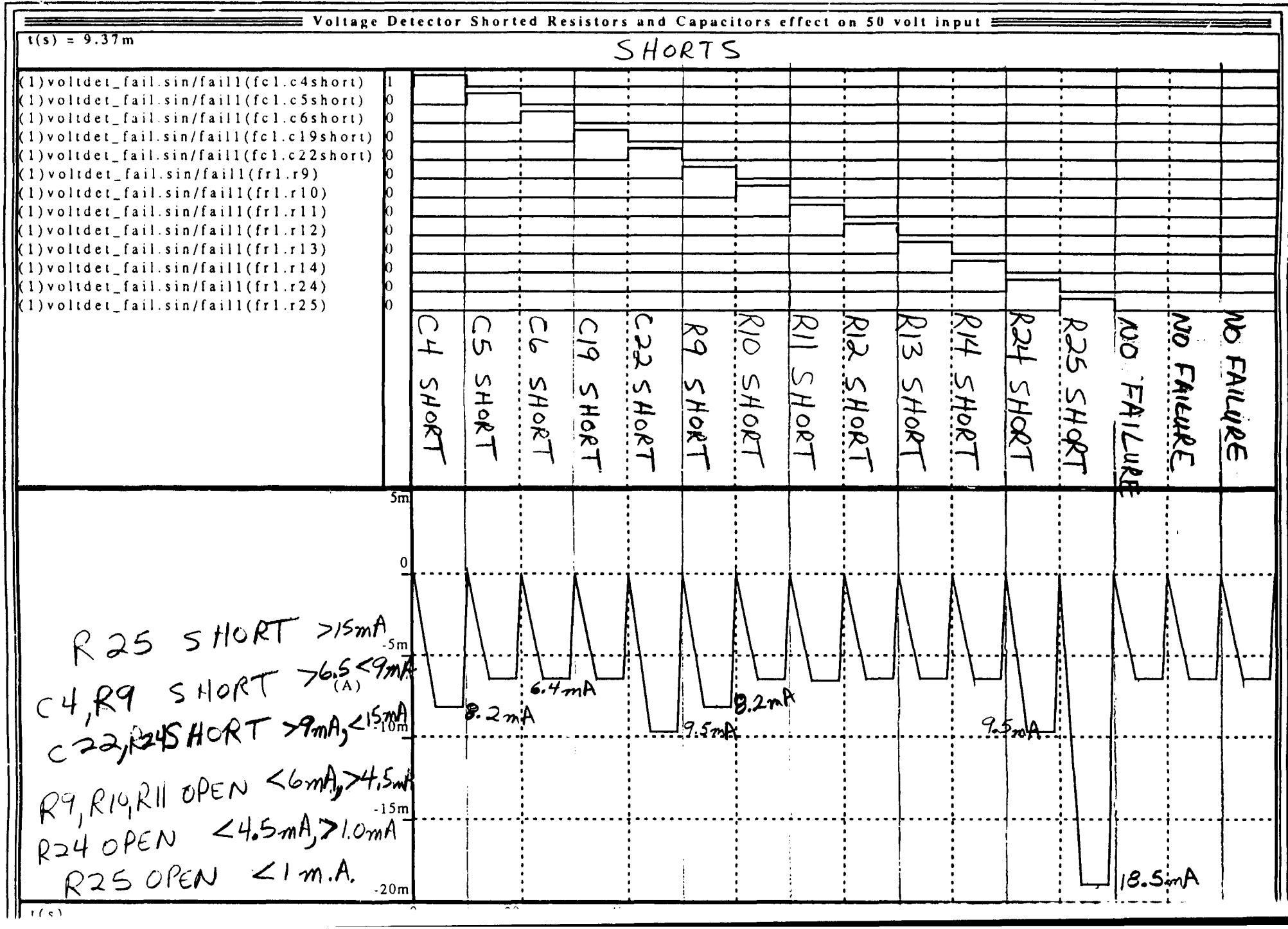
VOLTAGE DETECTOR SCHEMATIC, FIGURE 9.

GRAPHICAL FAULT DICTIONARY

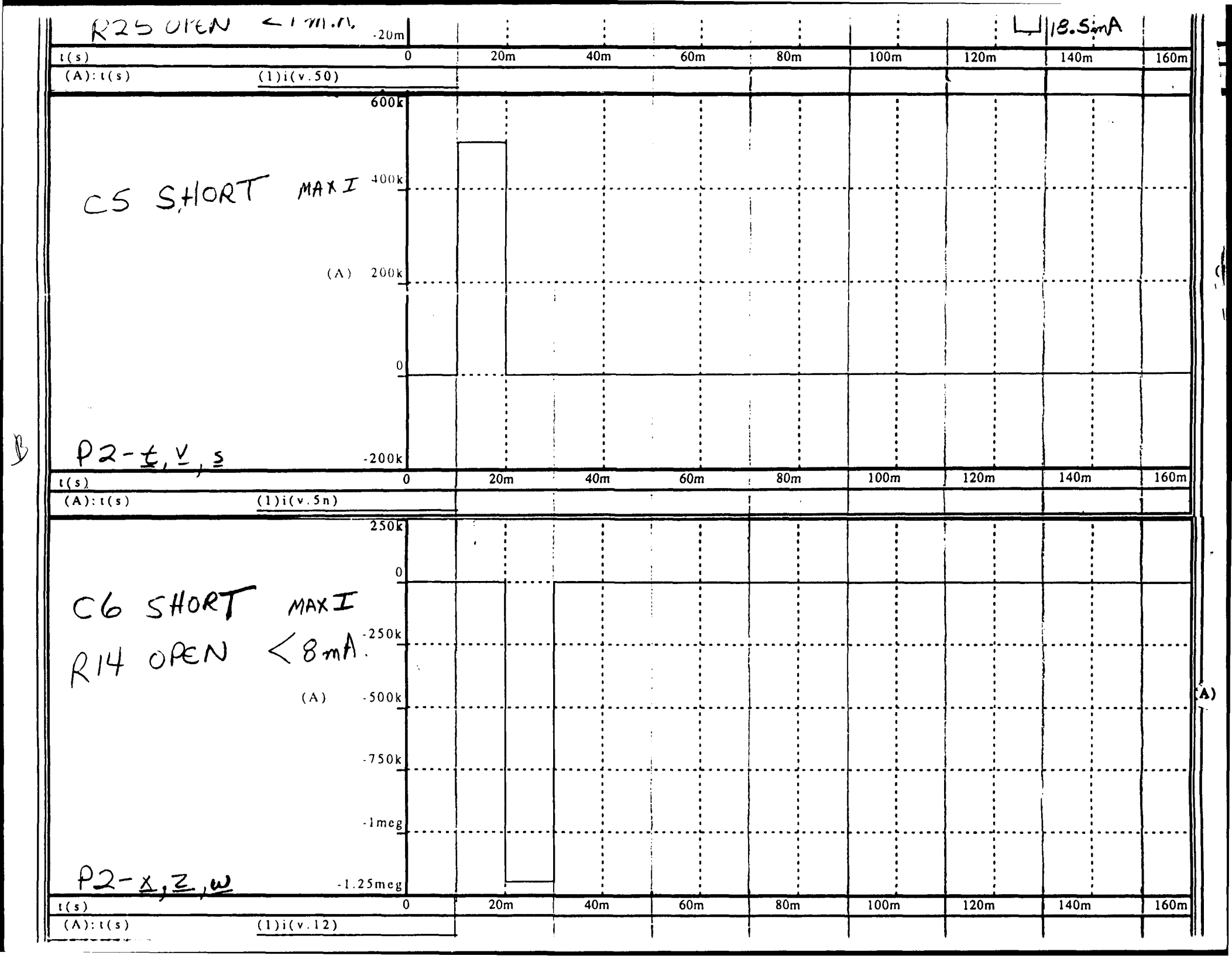
FIGURE 10

GRAPHICAL FAULT DICTIONARY
FIGURE 10

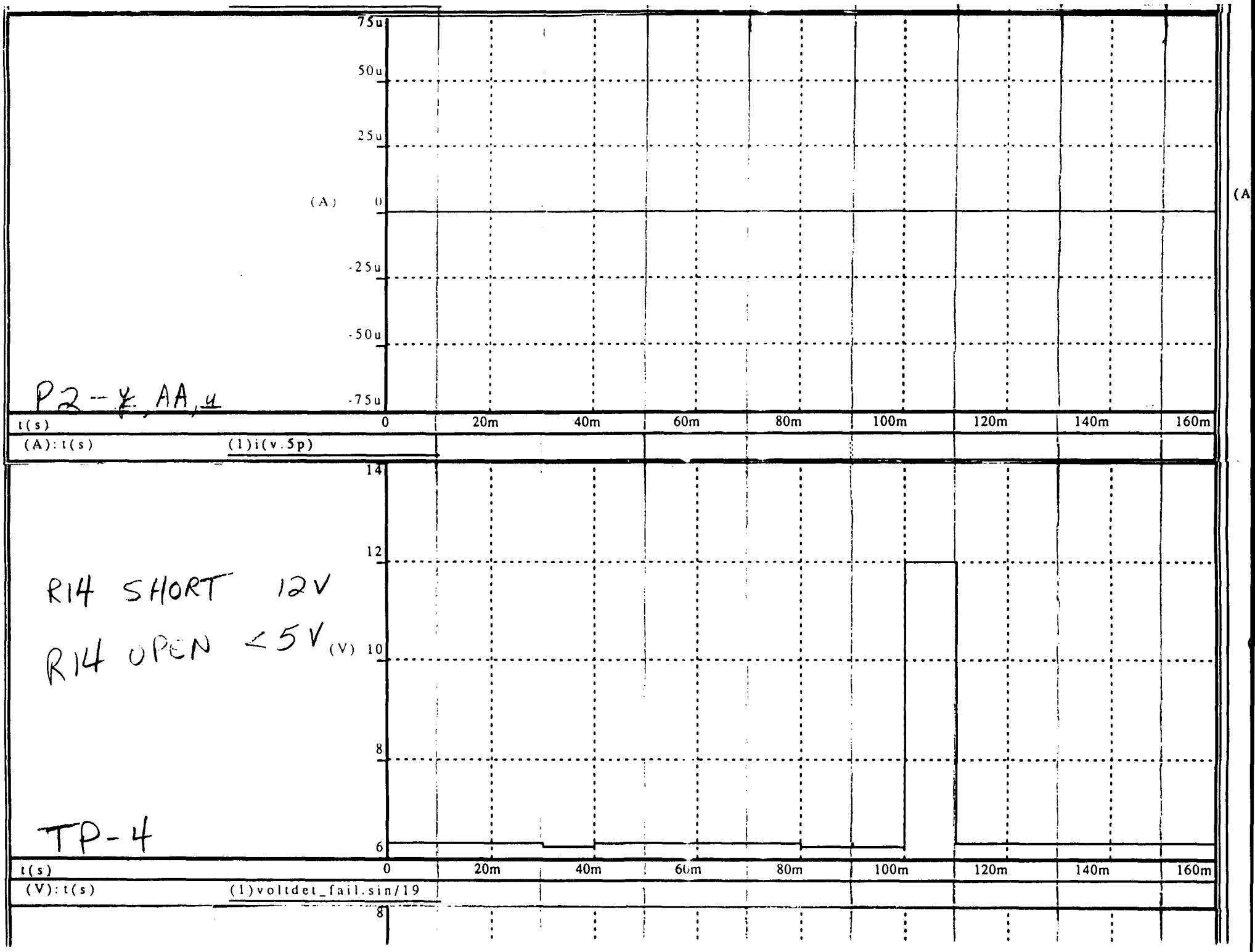
LEFT SIDE #1



A


R25 OPEN < 1 m.A.
-20m
18.5mA
t(s)
0
20m
40m
60m
80m
100m
120m
140m
160m
(A):t(s)
(1)i(v.50)
600k
400k
200k
0
-200k
C5 SHORT MAX I
(A)
P2-t, V, s
(A):t(s)
(1)i(v.5n)
250k
0
-250k
-500k
-750k
-1meg
-1.25meg
C6 SHORT MAX I
R14 OPEN < 8mA.
P2-x, z, w
(A):t(s)
(1)i(v.12)

LEFT SIDE #2

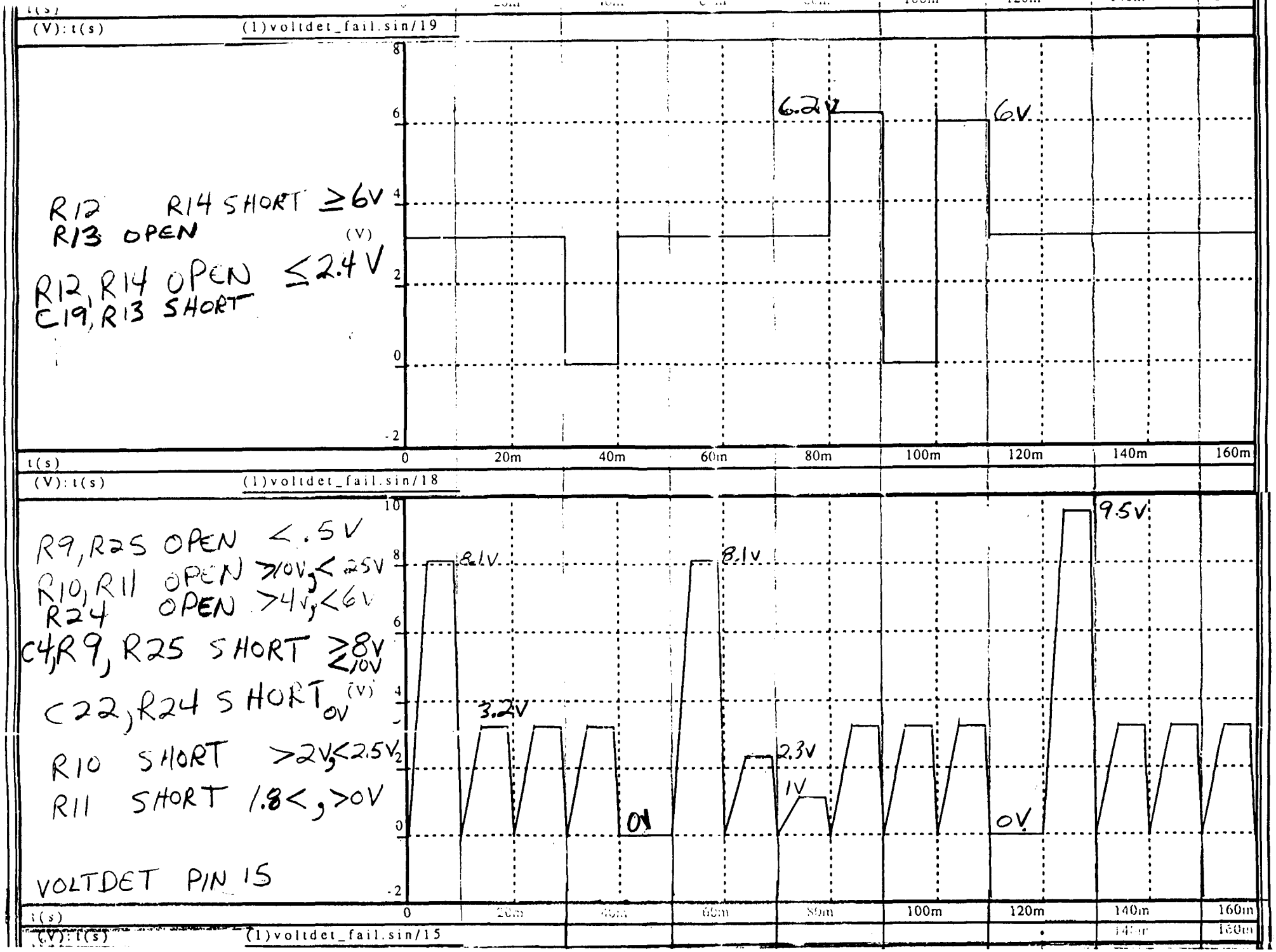


P2 -- ~~X~~, AA, ~~4~~

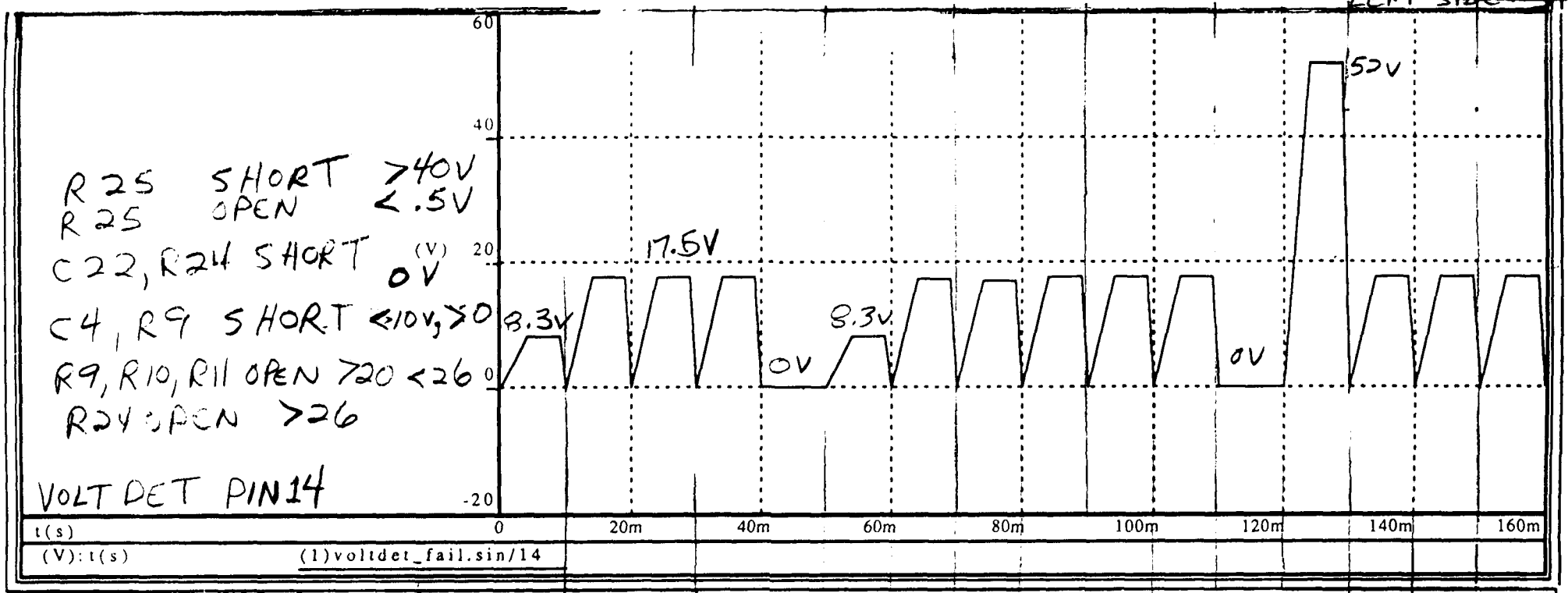
R14 SHORT 12V

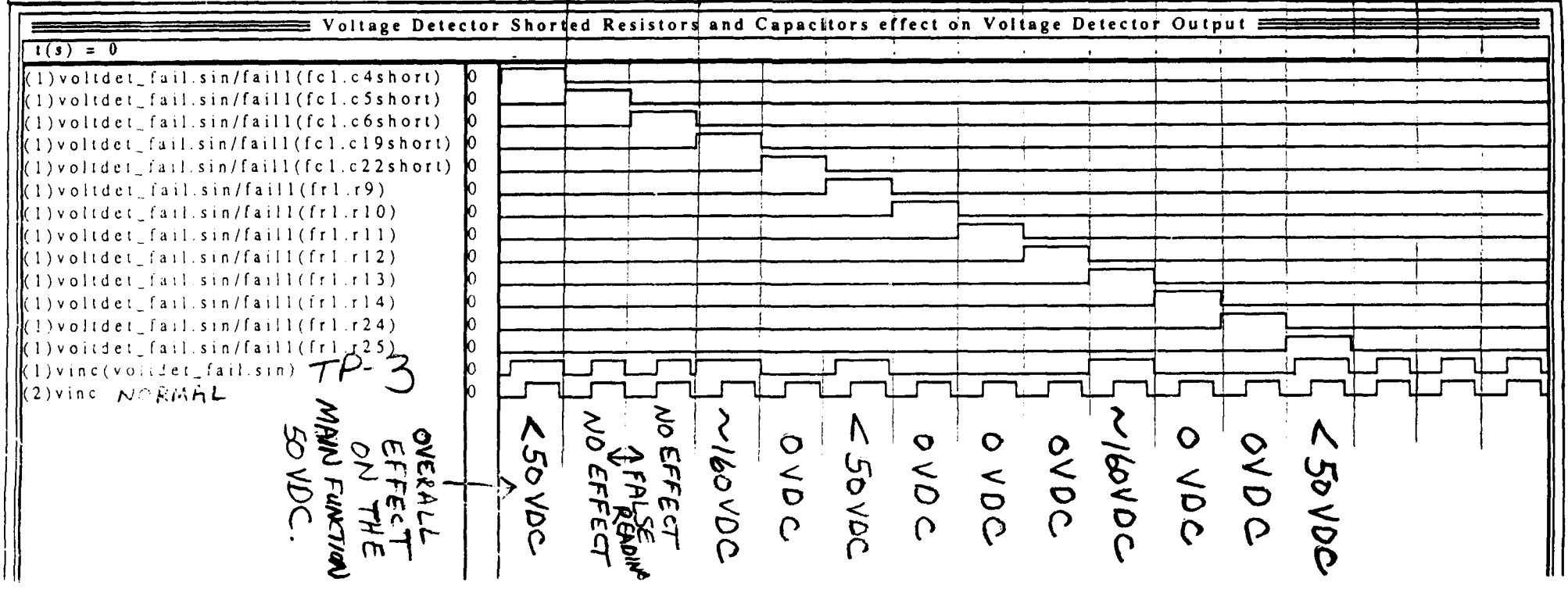
R14 OPEN < 5V

TP-4


(V):t(s)
(1)voltdet_fail.sin/19
R12 R14 SHORT ≥6V
R13 OPEN
R12, R14 OPEN ≤2.4V
C19, R13 SHORT
6.2V
6V
(V)
8
6
4
2
0
-2
t(s)
0
20m
40m
60m
80m
100m
120m
140m
160m
(V):t(s)
(1)voltdet_fail.sin/18
R9, R25 OPEN <.5V
R10, R11 OPEN >10V, <25V
R24 OPEN >4V, <6V
C4, R9, R25 SHORT ≥8V ≤10V
C22, R24 SHORT 0V
R10 SHORT >2V, <2.5V
R11 SHORT 1.8<, >0V
VOLTDET PIN 15
8.1V
8.1V
9.5V
3.2V
2.3V
1V
0V
0V
10
8
6
4
2
0
-2
t(s)
0
20m
40m
60m
80m
100m
120m
140m
160m
(V):t(s)
(1)voltdet_fail.sin/15

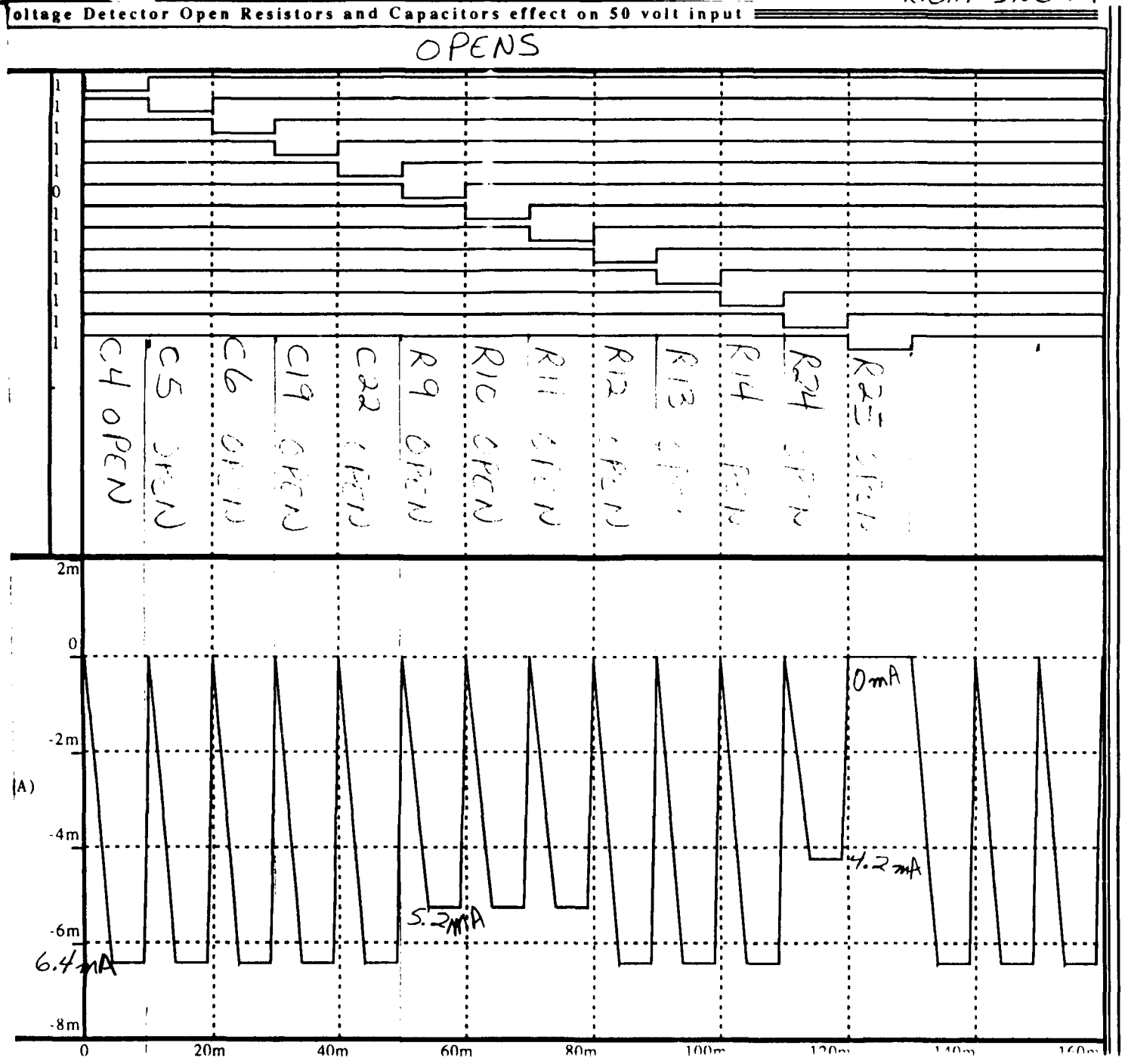
LEFT SIDE #2

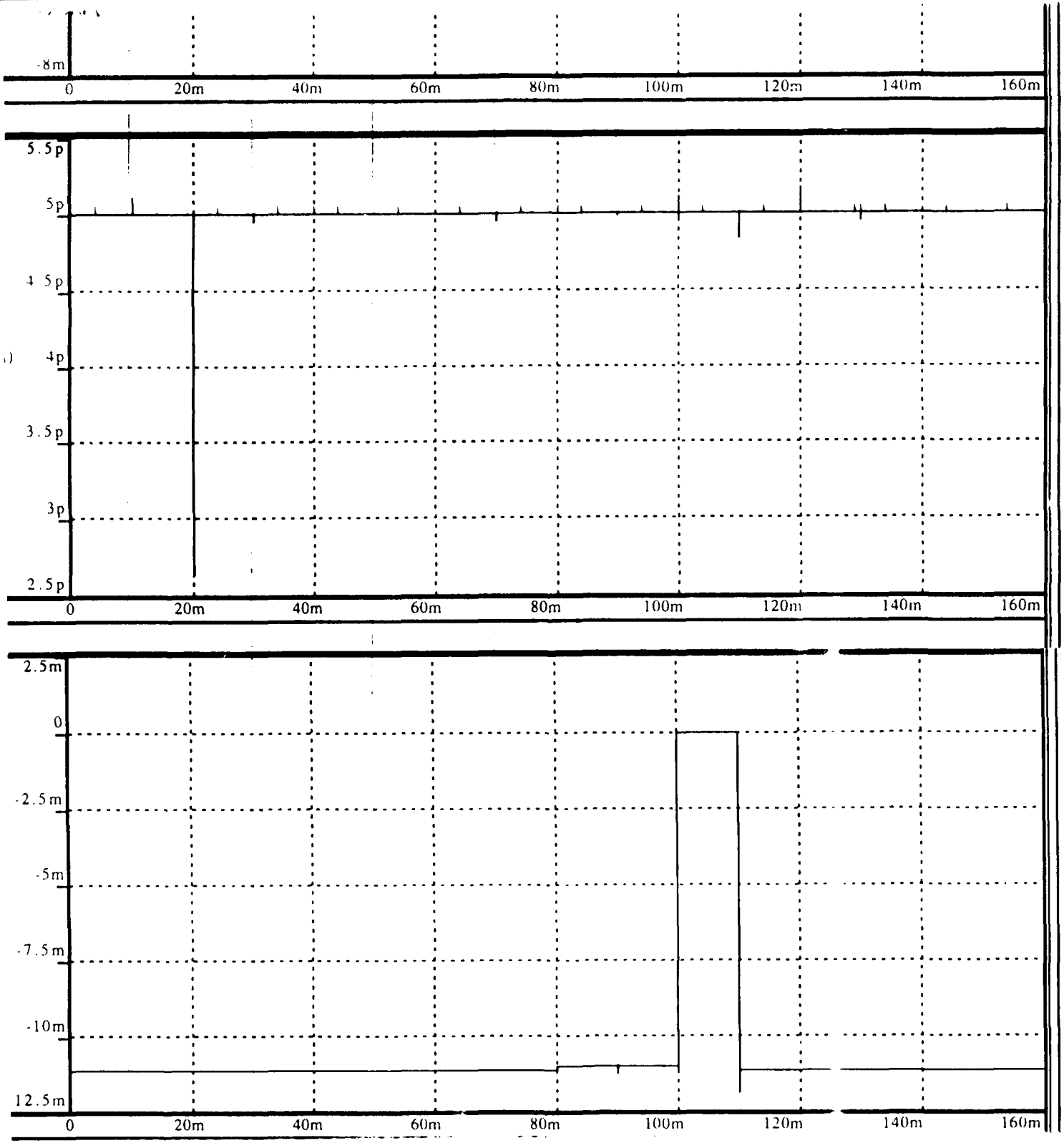

R25 SHORT >40V
R25 OPEN <.5V
C22, R24 SHORT 0V
C4, R9 SHORT <10V, >0
R9, R10, R11 OPEN 720 <26
R24 OPEN >26
VOLT DET PIN 14
(V)
60
40
20
0
-20
17.5V
8.3V
0V
8.3V
0V
52V
t(s)
0
20m
40m
60m
80m
100m
120m
140m
160m
(V):t(s)
(1)voltdet_fail.sin/14


Voltage Detector Shorted Resistors and Capacitors effect on Voltage Detector Output
t(s) = 0
(1)voltdet_fail.sin/fail1(fc1.c4short)
(1)voltdet_fail.sin/fail1(fc1.c5short)
(1)voltdet_fail.sin/fail1(fc1.c6short)
(1)voltdet_fail.sin/fail1(fc1.c19short)
(1)voltdet_fail.sin/fail1(fc1.c22short)
(1)voltdet_fail.sin/fail1(fr1.r9)
(1)voltdet_fail.sin/fail1(fr1.r10)
(1)voltdet_fail.sin/fail1(fr1.r11)
(1)voltdet_fail.sin/fail1(fr1.r12)
(1)voltdet_fail.sin/fail1(fr1.r13)
(1)voltdet_fail.sin/fail1(fr1.r14)
(1)voltdet_fail.sin/fail1(fr1.r24)
(1)voltdet_fail.sin/fail1(fr1.r25)
(1)vinc(voltdet_fail.sin) TP-3
(2)vinc NORMAL
OVERALL EFFECT ON THE MAIN FUNCTION 50 VDC.
<50 VDC
NO EFFECT
FALSE READING
NO EFFECT
~160 VDC
0 VDC
<50 VDC
0 VDC
0 VDC
0 VDC
~160 VDC
0 VDC
0 VDC
<50 VDC

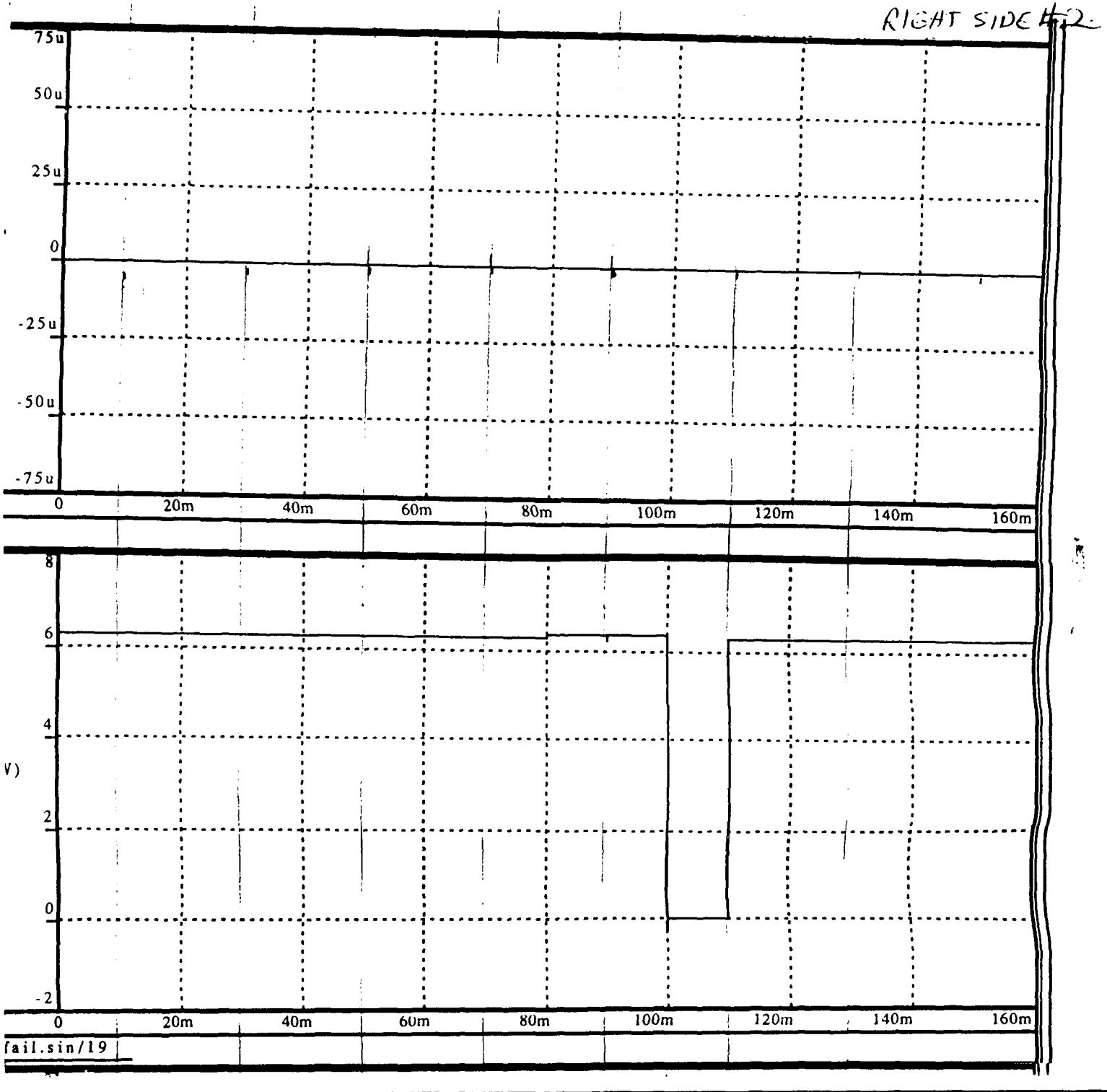
GRAPHICAL FAULT DICTIONARY
FIGURE 10

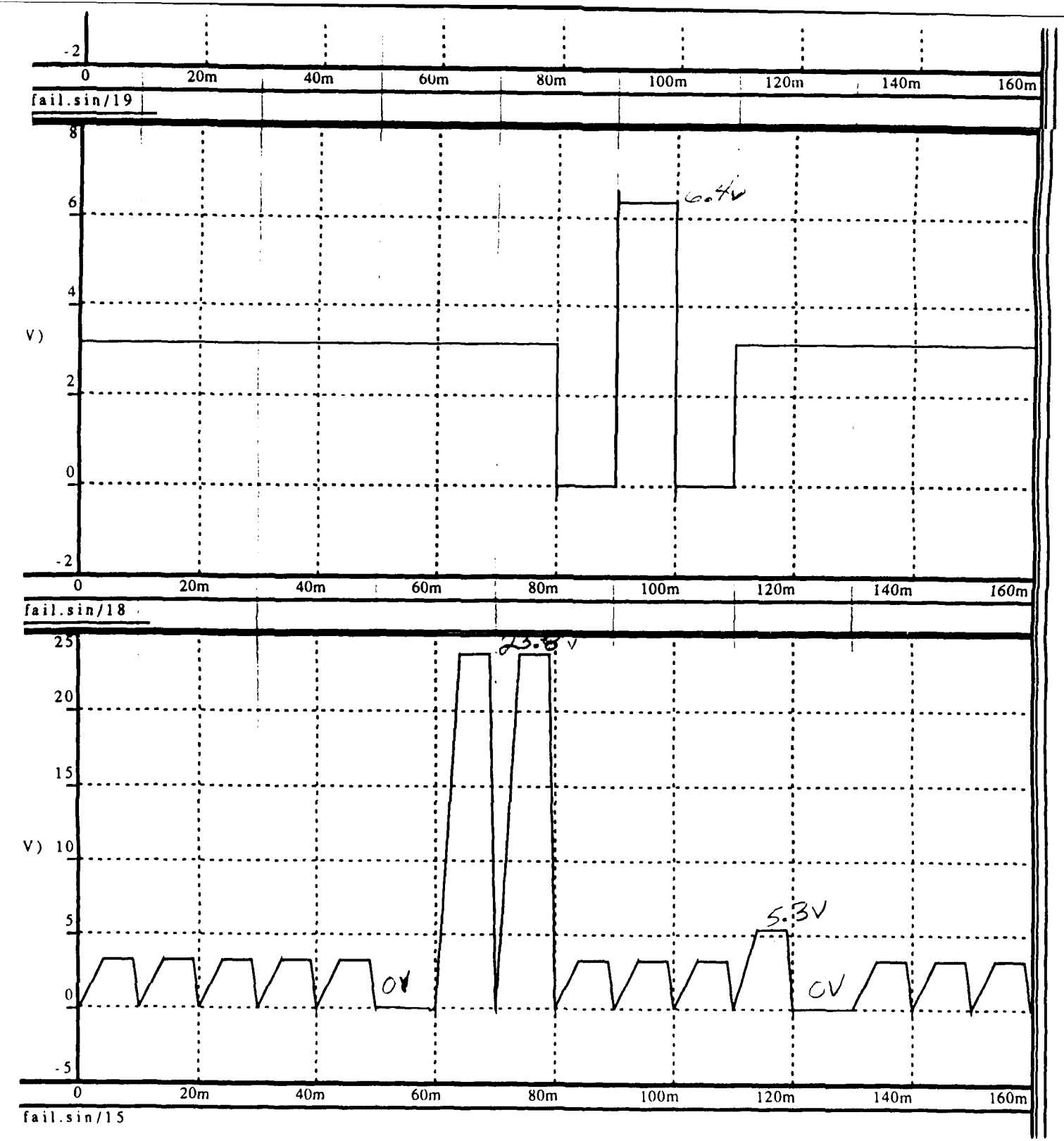
RIGHT SIDE #1



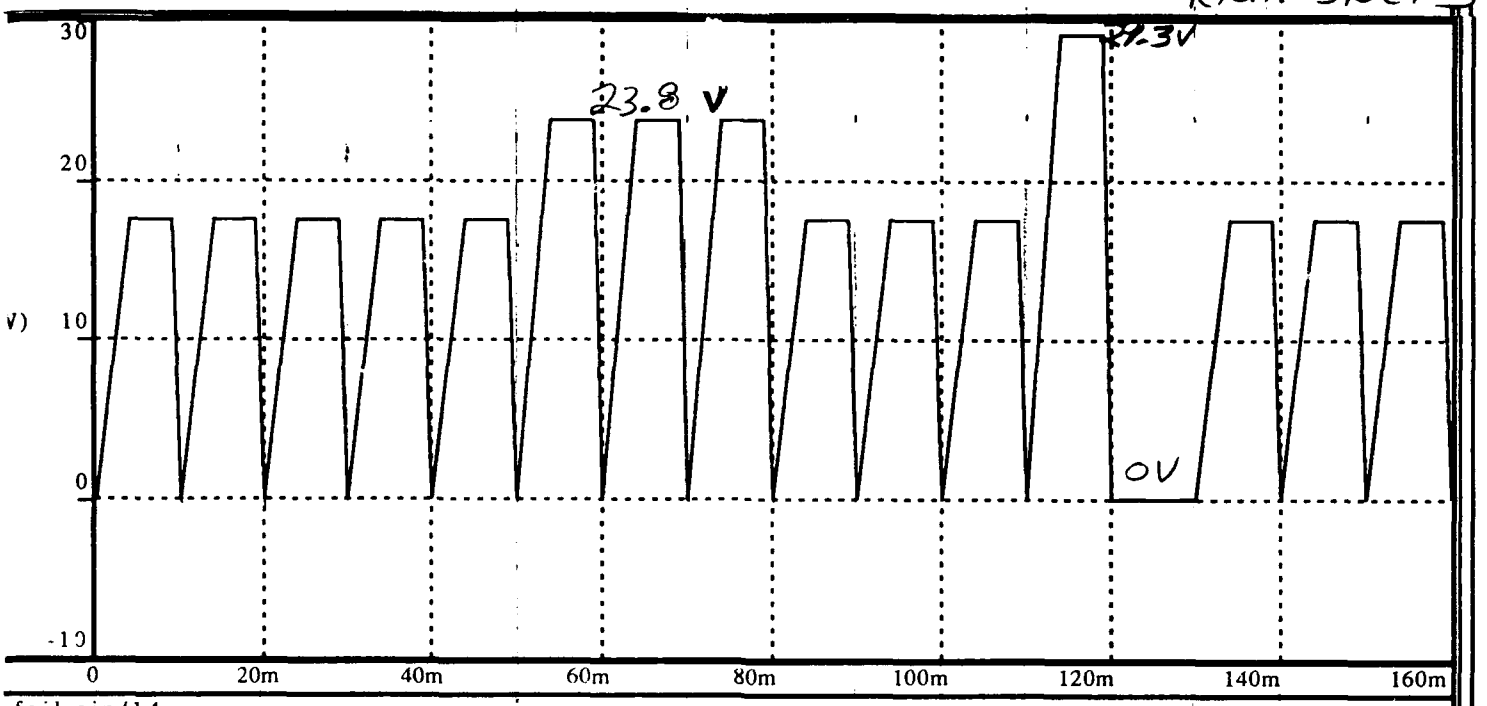

-8m
0
20m
40m
60m
80m
100m
120m
140m
160m
5.5p
5p
4.5p
4p
3.5p
3p
2.5p
0
20m
40m
60m
80m
100m
120m
140m
160m
2.5m
0
-2.5m
-5m
-7.5m
-10m
12.5m
0
20m
40m
60m
80m
100m
120m
140m
160m

RIGHT SIDE 42



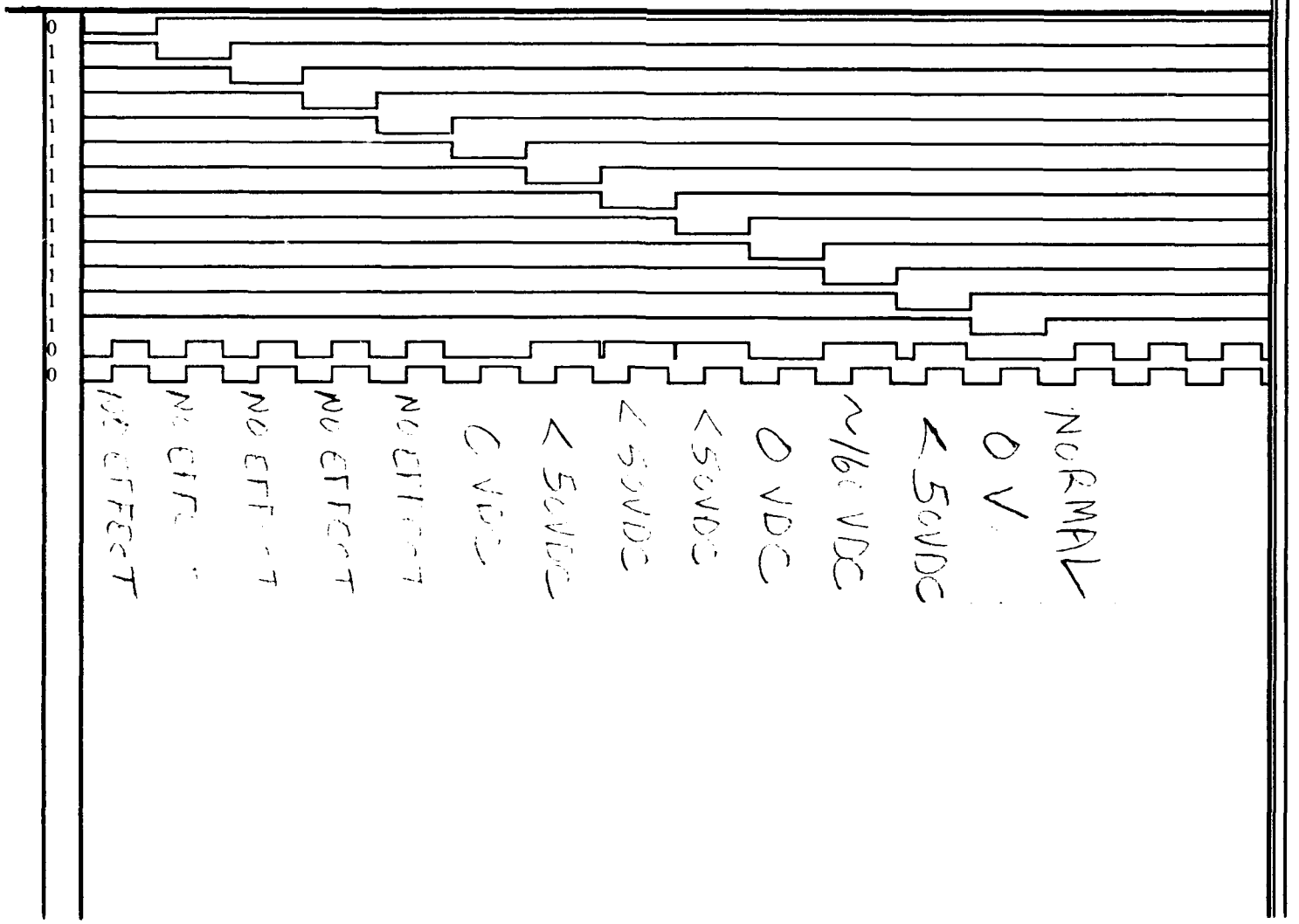

-2
0
20m
40m
60m
80m
100m
120m
140m
160m
fail.sin/19
8
6
4
V)
2
0
-2
6.4V
fail.sin/18
25
20
15
V) 10
5
0
-5
23.8V
5.3V
0V
0V
fail.sin/15

RIGHT SIDE #3



fail.sin/14

Fault Dictionary for Open Resistors and Capacitors of the Voltage Detector



The faulted component and component failure mode is annotated on the top horizontal section for each fault time bin. A fault state is identified when the same voltage level exists for different component fault modes. The fault state identifies the failure modes which present the same fault symptoms. A group of components presenting the same fault symptoms is called a fault set. An ambiguity group is derived from fault sets. Fault sets corresponding for each fault state are annotated on the right side of the Graphical Fault Dictionary. A complete fault component set is derived from a collective and comparative analysis between a fault symptom and all possible faults. Not all possible fault modes are displayed in the illustrated Graphical Fault Dictionary. Individual tests of the testing sequence identifies the fault state electrical parameters. Fault state electrical parameterics are defined as the range of measurable parameters (voltages, frequencies, etc.) that describe the fault symptom (i.e. fault state). The tests are written to the right of the fault sets on the diagram.

The testing information generated by analysis of the Graphical Fault Dictionary is recorded as an entry in a fault dictionary. An entry in a fault dictionary relates a set of possible faults for a given fault symptom under given testing conditions. The fault dictionary is a text table relating faults to fault symptoms. Each line entry of the fault dictionary is a fault state. A fault state is a range of electrical parameters (e.g. voltages) that describe the faulted condition or symptom. The fault dictionary structure is illustrated in Figure 11.

7.5 FAULT DICTIONARY ANALYSIS

Fault dictionary analysis is the processing and extraction of the fault dictionary information to develop ambiguity groupings, diagnostic testing tolerances, and diagnostic test sequences.

Ambiguity groups are calculated from the union set formed with multiple fault sets or are the individual fault sets in the fault dictionary. Figure 12 illustrates this concept and ambiguity calculation for the studied Voltage Detector.

Diagnostic testing tolerances are determined by Monte Carlo fault simulation for the fault or set of faults being simulated. Monte Carlo analysis as a test point establishes the average, minimum and maximum electrical values. If the differences between fault states or symptoms are distinct and because Monte Carlo simulation is resource intensive, an alternative approach is to set testing tolerance guard bands at plus or minus 20 per cent of the difference between fault symptoms. For example, if fault symptoms are 0 volts and 10 volts at a test point, then the testing tolerance for the 10 volt fault grouping would be set at 8 volts for the lower test limit.

FAULT DICTIONARY STRUCTURE

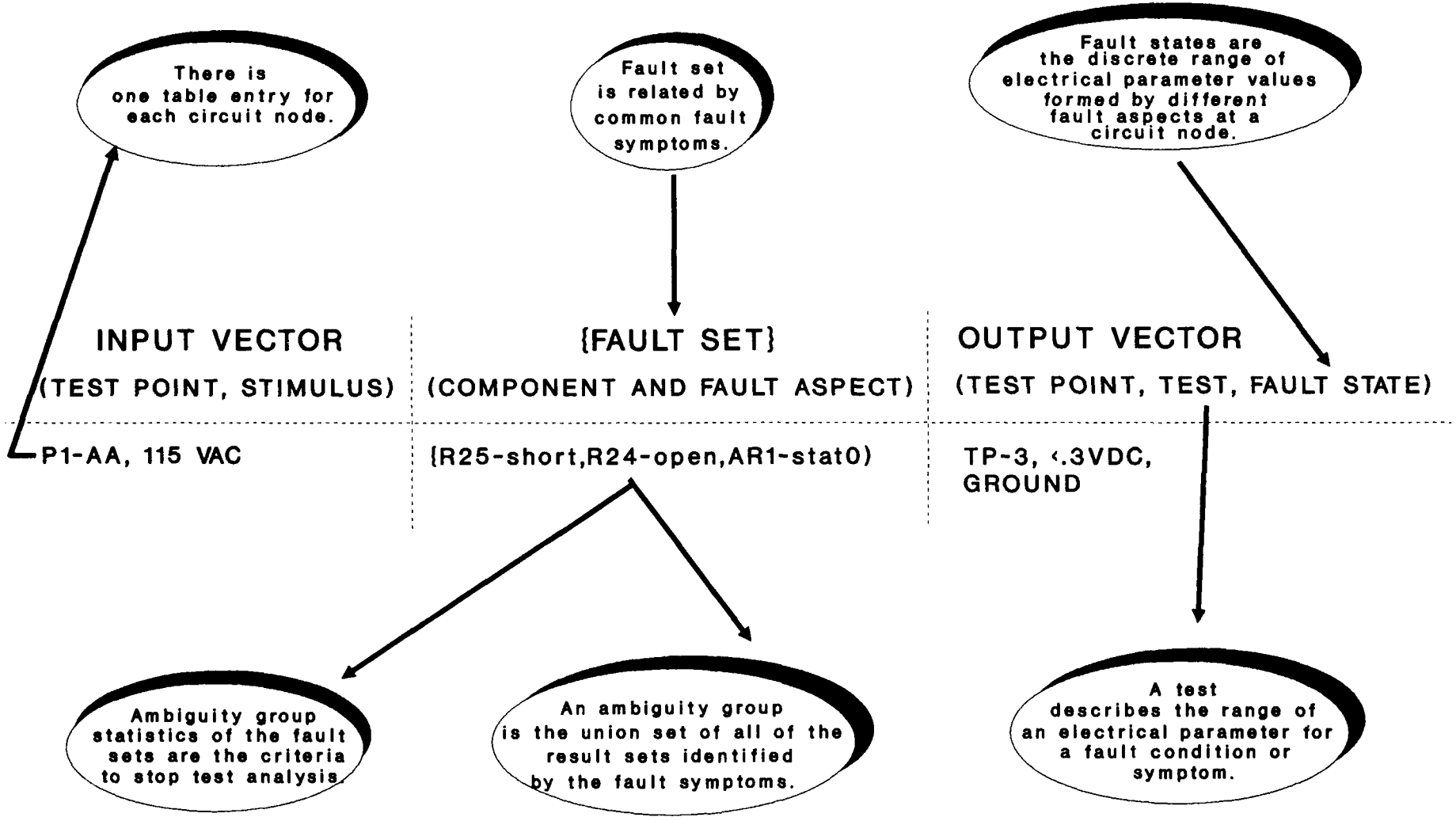


FIGURE 11

Ambiguity Groups for the Voltage Detector Function

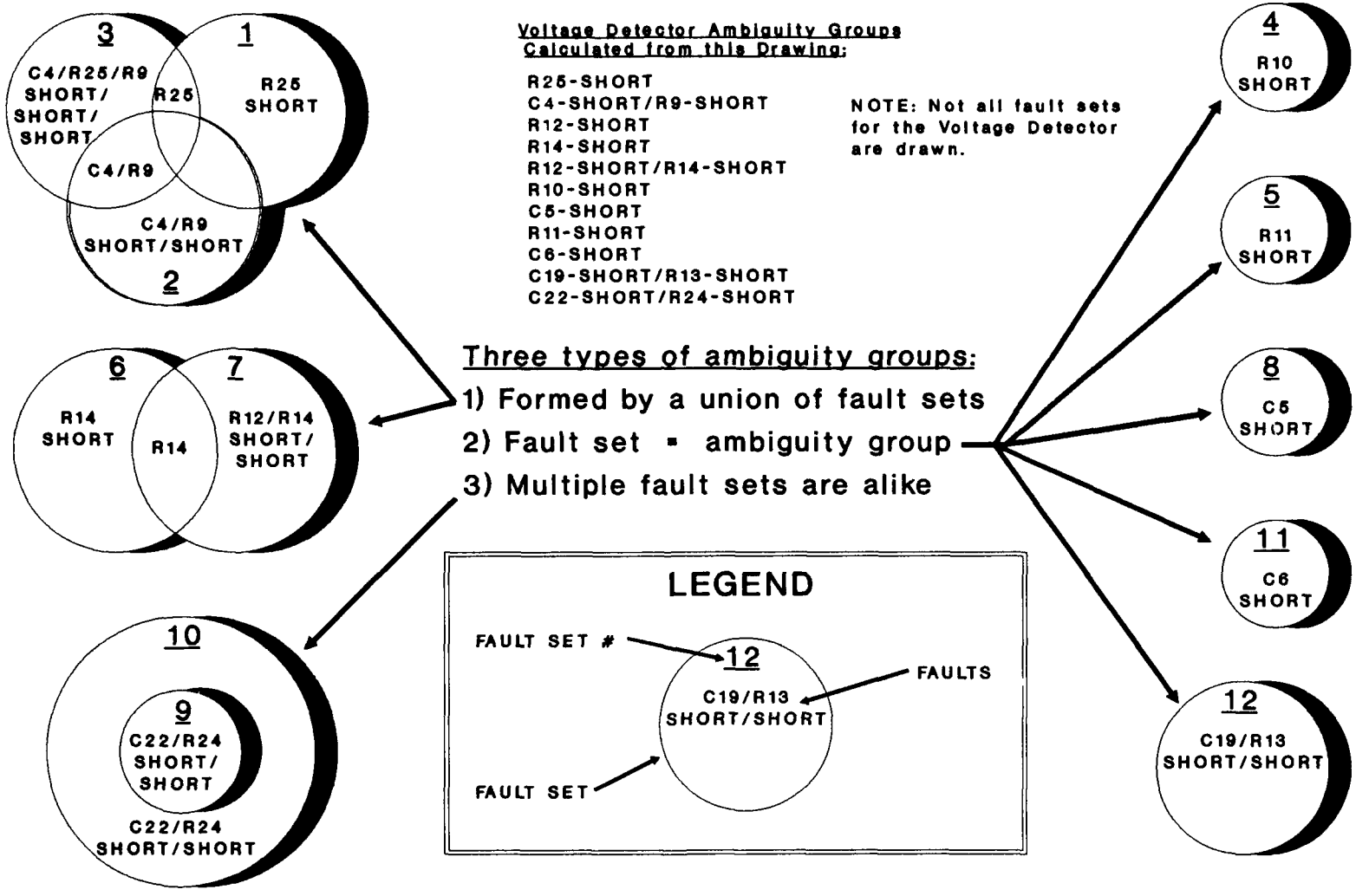


FIGURE 12

Diagnostic testing sequence is a derivative product form of the ambiguity group analysis. After the ambiguity group statistics are approved, the diagnostic testing sequence is derived from the individual fault set tests relating to the ambiguity group. For example, the number 1,2, and 3 fault sets each have a test associated with them and are required to isolate the fault to the R25 and C4, R9 ambiguity group. These individual tests are placed in the Voltage Detector functional subsection of the diagnostic program. The test ordering is in a sequence to reduce the faults in the union set formed with the remaining fault sets. This mathematical procedure defines a logical fault isolation testing sequence to identify an ambiguity group. The end goal is to generate a diagnostic procedural flow chart. The diagnostic procedural flow chart plus diagnostic testing tolerances and input testing vectors form the diagnostic flow chart which provides a diagnostic skeleton for the ATLAS program.

Testing strategy involves what and how to test a function during STTO, end-to-end, diagnostic, and alignment tests. Testing strategy is determined from the available TPS documentation, the engineers experience, and his functional and operational circuit knowledge.

Diagnostic testing strategy covers many different techniques, one of which based on the fault dictionary approach. The main task is to select a combinational series of input testing vectors which completely tests all functional and operational circuit modes. Also, part of the strategy task is to decide what parameters are important to test: voltage, current, resistance, frequency response for end-to-end testing and diagnostic testing.

7.6 TESTABILITY PARAMETERS

Important testability parameters are ambiguity group size and distribution. Testability analysis is an iterative process to decide the optimum ambiguity size and distribution when the specified fault isolation level cannot be achieved because of real world constraints. Real world constraints require tradeoffs between number of un-detects/non-detects and number of test steps and number of probable test points. Limited TPS resources is another constraint. Theoretically, there are no non-detects because all components contribute to the functionality of the SRA. Probable test points, physical constraints, temperature, or overcurrent limitations in application of test are examples of physical limitations.

Calculation of ambiguity group statistics is a mathematical exercise utilizing the fault sets in the fault dictionary. The fault sets are converted into ambiguity group sets. The number of faults in each ambiguity group and the number of ambiguity groups are counted. These two

preliminary calculations provide the raw numbers for the relative and cumulative distribution calculation. The comparison of calculated relative and cumulative statistics with the SRA and ID ambiguity group size specifications determines if the testability task is complete.

8.0 WSTA

The Weapon System Testability Analyzer (WSTA) program analyzes and determines testability of a new weapon system design. WSTA's advantages are the number of testability analysis parameters generated and a testing sequence optimized for TPS runtime, repair cost or isolation cost tradeoff factors. WSTA's disadvantages are the difficulty in assimilating, acquiring, and the level of effort required to generate the desired testability information. The disadvantages are described individually in the following paragraphs.

Assimilating the required WSTA input information into one place is a problem. WSTA's input information of fault dependency data, logistical data, and schematic data has to be available in electronic form due to the large volume and costs of entering the data. Acquiring the data through a VHDL (Very large scale integration Hardware Description Language) and LSA (Logistical Support Analysis) computer file is a technical problem, yet to be worked out.

Acquiring a VHDL circuit description is a significant problem. The circuit under study first must be drawn with a CAE schematic program, then a circuit representation is output as a VHDL netlist. Using VHDL for WSTAs input requires remodelling of the circuit. Three port components (eg. transistors) need to be remodelled as a combination of two port components. The VHDL circuit information is translated into WSTA's internal database by a CAE preprocessor. WSTA's CAE preprocessor describes all component fault failures as an open circuit and not as a component short failure or any other fault mode. For an average SRA of 300 components, manual entry and dependency analysis of an average 600 failure modes are required, thus manual entry of the other fault failures(besides an open) is a significant effort. A program to directly insert the fault aspects into the WSTA's database is technically possible, but program development costs time and money.

The level of effort to generate the fault dependency data, model the circuit and enter the fault aspects is at odds with the value of information gained for testing strategy purposes. The fault dependency data, in a readily compatible form is not available in traditional TPS development. Generation of fault dependency data is possible through the process described in Section 7.0 or through traditional technical analysis means.

WSTA is a useful tool for testability analysis after fault dependencies are created and input into WSTA. If the information is in a form readily transferable into WSTA, then the testing sequence and ambiguity results for TPS development warrant the level of effort expended. However, the required WSTA inputs either do not exist, or the inputs are in a form which are not immediately importable.

9.0 FUTURE CIRCUIT SIMULATION TASKS

Good database structure and database access facilities are required to improve the efficiency and productivity of CAE software for TPS development. User and programming development of the native capability is needed to fully realize the CAE environments full potential for TPS development. A TPS development simulator, SVS (Software Verification Simulator) design rule checking, and TPS design management functions are proposed capabilities to be added to the CAE environment.

9.1 TPS DEVELOPMENT SIMULATOR

The TPS development simulator simulates all operational parts to a TPS which entails the ATLAS software, CASS resources, ID and the UUT. The ATLAS software and CASS test instruments are simulated behaviorally in a functional sense. The ID and UUT are simulated at the component level. The purpose of the simulator is to reduce integration time through simulation of the CASS tester. Two specific purposes are to do design verification testing (to ensure the software, CASS, and ID "play" together), and to test the software detection of a fault. The justification for a TPS development simulator is to make the test program and ID design a dynamic and highly iterative process which ensures an efficient, productive and quality TPS product. The simulation provides the answers for the **CRITICAL** TPS integration questions:

1. Does the test program find the fault?

2. Does the ATE via the test program and ID provide the correct stimulus to the UUT?

3. Does the ID and ATE test program measure the correct response from the UUT?

The ATLAS simulation model is created by the TPS engineer writing an ATLAS simulation model during TPS development or by use of an ATLAS interpreter and modelling program. When the TPS engineer is satisfied with the correct ATLAS software simulation, an ATLAS compiler generates the ATLAS testing code for the diagnostic and end-to-end program

TPS DEVELOPMENT SIMULATOR

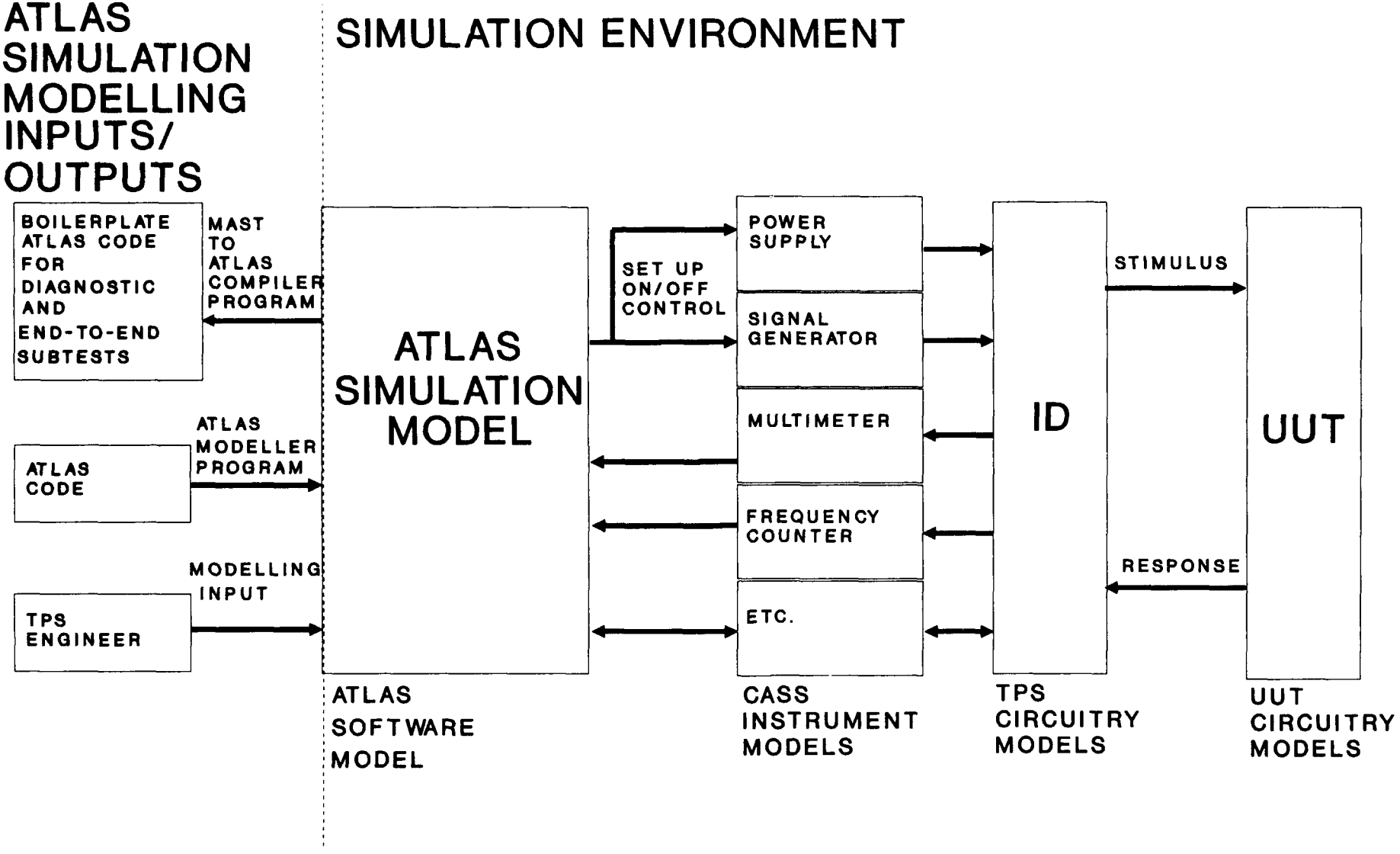


FIGURE 13

subtest sections. The ATLAS interpreter and modelling program creates an ATLAS simulation model from existing ATLAS code. Figure 13 is a block diagram illustrating the different models and the models relationship in the simulation. First the ATLAS code is modelled by a person or a syntax interpreter and ATLAS modeler program. The CASS instrument models, ID, and UUT models previously exist from the TPS development effort. The TPS model is simulated and the results are analyzed for correctness.

Several technical approaches for a TPS development simulator are possible. Each approach consists of different levels of refinement and levels of effort to achieve the objective. Refinement levels refer to the modelling detail of the CASS tester. The electrical characteristics of the switching paths need not be modelled with the exception of non-RF or high current situations. Basic functional operation of the CASS instruments are modelled. For example, a DC power supply is a DC source with a series resistor. Setup and control of the DC source or any CASS instrument is easily accomplished with Saber's analog modelling programming language. The analog modelling programming language (MAST) and the analog simulator (Saber capabilities are the CAE specific capabilities) enable the TPS development simulator concept to come to fruition.

The TPS development simulator combines the normally separate TPS design tasks into a common effort. The TPS software and hardware are simulated together. Simultaneous software and hardware simulation completely changes the TPS development methods from a static to a dynamic process. The ATLAS and ID design is accomplished simultaneously during an iterative design process. Integration errors are discovered and corrected by the simultaneous simulation. Discovery of ID and ATLAS design errors in a design phase reduces the cost of fixing the design errors which are normally discovered during integration (after an ID prototype is built and the ATLAS code is written). Typically, an ID prototype is rebuilt and the ATLAS code is completely rewritten from the initial design effort. The cost of correcting design errors at integration time is ten times the cost of correcting the design error during the previous design phase. This is why computer simulation during design is so valuable to the electronic design industry.

A working model and proof-of-concept for the TPS development simulator is proposed. A section of ATLAS code would be translated by an engineer into an equivalent MAST simulation model. Also, rudimentary models of the basic CASS instruments would be created. The ID and UUT would be modelled at the component level. Simulation of the software, CASS, ID and UUT models is required to prove the technical feasibility. The effort would take two months to accomplish and require a workstation, and a schematic capture program with the SABER analog simulator.

9.2 SVS DESIGN RULE CHECKING

The System Verification Simulator (SVS) verifies or checks for pinout errors, wiring errors and UUT I/O functional and electrical requirements. There are general electrical rule checks (eg. for ground shorts or open circuits), design rule checks (eg. for signal type (digital, analog, input, output), compatibility checks, and specification checks (over/under voltage or current). Electrical and design rule checks exist as part of an enhanced schematic capture program. Not all schematic capture rule checkers are configureable, nor do they inform the engineer of a misconnect error. A misconnect error is found when the design violates an electrical, design, or specification check. However, a misconnect error may pass all the aforementioned checks but will be discovered during simulation. In a simulation, the misconnect is discovered because the circuit does not display the expected functional behavior because it is connected to the wrong ID or UUT input or output pin.

A specification checker does not exist as a schematic feature. A specification checker compares an electrical parameter specification against an actual electrical parameter at a specified input or output pin. This feature is possible, if you know:

1. What the electrical specifications are for an ID or UUT pin.

2. What are the actual electrical parameters applied to the ID or UUT pin.

For the SVS, the actual electrical parameters are derived from the ATLAS program directly, assuming no active ID circuitry. If active signal conditioning circuitry exists in the ID, then analysis of simulation results provides the actual electrical parameters.

Addition of design and specification rule checking capabilities are proposed as part of the TPS development simulator to assist the TPS engineer during TPS development.

9.3 TPS DESIGN MANAGEMENT FUNCTIONS

Much of the TPS design information is organized or related to the repairable component. Ambiguity groups, non-detectable/un-detectable components, pin voltages or current specifications, pin simulation results, and MTBF results are TPS design parameters related to a component. The component database generated as a result of schematic entry provides a data structure to organize TPS design parameters. Extra data fields added to a component's parameter database allows for inclusion of component related parameters into the UUT

circuit design database. TPS design parameters are generated during the testing analysis task and are used in many different TPS design calculations. The TPS design calculations are involved in the fault modelling, fault dictionary, fault dictionary analysis, and testability parameters. Fault modes (e.g. opens and shorts) are added to a component's database field during fault modelling. A components fault mode database field then configures the fault simulation for the fault mode under simulation. Fault dictionary parameters, such as fault sets and input or output electrical parameters are stored in the components probable test point and electrical device components database. Fault analysis calculations on the fault dictionary generates the ambiguity groupings. Testability parameters are a relative and cumulative statistical calculation of the ambiguity groupings.

The development of the TPS design management functions requires the addition of the TPS design parameters into a component's database, and the creation of processing routines to extract and process the component data to perform ambiguity group, testing criteria, and fault dictionary calculations.

10.0 CONCLUSIONS

Results of the CAE study, to date, reveal great potential in the application of CAE software for TPS development. The current raw capability exists in several CAE environments. However, a few months effort is required to configure the CAE environment for TPS development.

The recommended analog simulator for TPS development is SABER. SABER is the only analog simulator to perform analog fault simulation. Also, SABER possesses a sophisticated modelling language which allows electrical devices, mechanical devices, faulted components and software to be simulated as a system.

The report identifies CAE selection requirements. Selection of CAE software tools in a CAE environment is the next logical and natural step in the evaluation. Software integration of CAE programs and significant configuration and development efforts of the CAE environment are critical factors in the selection process. The CAE software tool selection is a three decision factor process. The decision processes are:

1. To identify the software tools to generate analog and digital fault data required for TPS development. Only one analog fault simulator has been identified, which is Saber. This fact sets the condition and constraints with selection of a CAE environment and software tools.

2. To select a CAE environment to accomplish the desired TPS CAE environment capabilities which further defines and restricts the selections.

3. If there is a choice between software tools, select the most mature and best integrated software.

The report studies an analog fault simulation approach to the TPS development process. The analog fault simulation approach provides great efficiency, productivity and quality improvements over traditional TPS development processes. The analog fault dictionary provides the essential data for the testing analysis phase of TPS development. Ambiguity groups and ambiguity group testing parameters calculations from the fault dictionary is possible. A complete fault simulation and fault analysis of the SRA is necessary to fully document the fault simulation approach for TPS development.

11.0 RECOMMENDATIONS

Recommendations are:

1. Develop a fault dictionary for the SRA under study. Translate the fault dictionary information into fault testing criteria, ambiguity callouts and diagnostic tests.

2. Narrow the CAE environment to TWO vendors. Rate the CAE environments on matching the desired CAE environment capabilities and the development effort (time and cost) to achieve the capabilities. Evaluate each CAE environment separately for a period of 30 days. Select a CAE environment. Setup and develop the CAE environments raw capability for TPS development.

3. Implement a demonstration of the TPS development simulator.

GLOSSARY OF TERMS

AMADEUS - Amadeus is the marketing product name for a standard set of software features, CAE programs and framework operating system environment produced by Cadence Design Systems. Amadeus includes a schematic drawing, design management, database library, and utility programs.

BEHAVIORAL MODELLING - Behavioral modelling is modelling of a components electrical behavior in simple or complex mathematical equations.

CADAT - CADAT is the name of a digital circuit and digital fault simulator. CADAT is used extensively for IC circuit design.

CALS - CALS is an acronym for the Computer Aided Logistical Support government program directives.

CGM - CGM (Computer Graphics Metafile) is a vector based drawing file format output from a graphics drawing program and input into desktop publishing programs.

COMPONENT MODELLING - Component modelling is the modelling of a component for a simulator. Also, component modelling is used to describe the very accurate modelling supplied by a company (ie. the native simulator model library).

DXF - DXF (Drawing eXchange Format) is a vector based drawing file format used to exchange drawings between software programs.

EDIF - EDIF (Electronic Data Interchange Format) is a file format for exchanging electrical circuit information (eg. netlist and symbol shapes) between CAE oriented software(eg. schematic drawing, simulator, and printed circuit board design programs).

HIERARCHIAL MODELLING - Hierarchial modelling models a function or assembly as a single substitutable part of a netlist rather than a collection of lower level components or assemblies.

HIERARCHIAL SIMULATION - Hierarchial simulation is a simulation of a hierarchial modelled SRA or WRA.

HPGL - HPGL (Hewlett Packard Graphics Language) is a drawing file format standard for Hewlett Packard plotters.

IGES - IGES (Inter Graphics Exchange Standard) is a vector based drawing file format standard for transferring drawings between software programs.

LSA - LSA (Logistical Support Analysis) is a government specification for recording logistical data.

MAST - MAST is the name of the Saber simulators' programming language.

MONTE CARLO - Monte Carlo is the random statistical component value selection process of Monte-Carlo simulation.

MONTE CARLO SIMULATION - Monte Carlo simulation is the multiple circuit simulation runs with different component electrical values.

PCB - PCB is an acronym for Printed Circuit Board.

SABER - Saber is an analog simulator produced by Analogy, Inc. Saber simulates analog and mixed mode circuits drawn in Amadeus.

SPICE - SPICE is the name of an analog simulator algorithm (ie. method of solving circuit equations). SPICE is an generic term describing analog simulators that employ the SPICE algorithm.

TEST VECTOR - A test vector is the input circuit stimulus required to test a circuit for a test response.

VERILOG - VERILOG is a digital simulator product of Cadence Design Systems.

VHDL - VHDL (Very large scale integration Hardware Description Language) is an acronym referring to the technology employed to manufacture an IC component.